

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-289144

(43)Date of publication of application : 27.10.1998

(51)Int.Cl.

G06F 12/00
G06F 12/16

(21)Application number : 09-092668

(71)Applicant : PIONEER ELECTRON CORP

(22)Date of filing : 11.04.1997

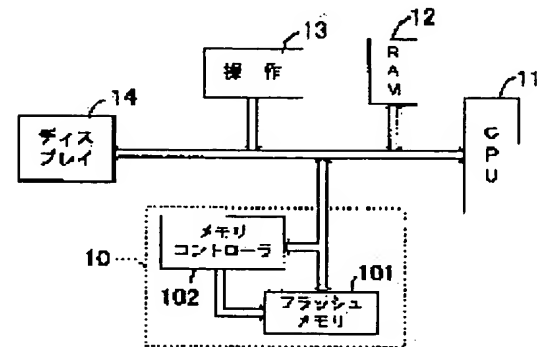
(72)Inventor : NAKAMURA TAKESHI

(54) MEMORY CONTROL METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To extend the service life of a flash memory by using first a sector that remains longest as a free area in terms of a time series when the free area is secured for writing the data.

SOLUTION: When the accesses are given to a storage 10 for the writing, erasion, etc., of files, a memory controller 102 controls a flash memory 101. In a file write mode, the FAT entries showing successively the idle sectors at and after the head of a FAT area are retrieved and the files are written into the sectors which are shown by the retrieved FAT entries. At the same time, the retrieved FAT entries are rewritten into the FAT entries showing the non-idle sectors. In a file erasion mode, the relevant file is deleted and also a new FAT entry is produced for the sector that becomes idle use to the file deletion. Then the new FAT entry is recorded at the end of the FAT area.



* NOTICES *

JPO and INPIT are not responsible for any damages caused by the use of this translation.

- 1.This document has been translated by computer. So the translation may not reflect the original precisely.
- 2.**** shows the word which can not be translated.
- 3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]

[Claim 1] It is the way each classifies a record section of a memory with two or more sectors which consist of predetermined storage capacity, and controls said memory by said sector unit, Record a fat entry which shows whether each of said sector is a vacant sector all over a fat field of said memory, and at the time of writing of a file. Write said file in a sector which searches a fat entry which shows that it is a vacant sector sequentially from a head of said fat field, and is shown by this search fat entry, and. Rewrite for a fat entry which shows that it is not a vacant sector about said search fat entry, and at the time of elimination of a file. A control method of

a memory deleting a file used as a candidate for deletion, and newly creating a fat entry about a sector used as a vacant sector, and recording this on the tail end of said fat field by deletion of said file.

[Claim 2]Heading sector information which shows a sector on which a fat entry which shows that it is a vacant sector among fat entries currently recorded all over said fat field, and exists in a very head position is recorded is recorded, A control method of the memory according to claim 1 searching a fat entry which shows that it is a vacant sector from a sector position shown using said heading sector information at the time of writing of a file.

[Claim 3]A control method of the memory according to claim 1 eliminating all the contents of record of said heading sector, and considering it as free regions which can write in this heading sector when it is shown that no fat entries currently recorded into a heading sector of said fat field are vacant sectors.

[Claim 4]A directory region where information which shows a position of a heading sector of file information about said files of each and said fat field is recorded on said memory, A control method of the memory according to claim 1, wherein record formation of the hook field where information which shows a position of a heading sector of said directory region is recorded is carried out.

[Claim 5]When a sector corresponding to said hook field turns into a bad sector, By recording the contents of record which information which searches a vacant sector out of said fat field, and shows a position of this vacant sector was recorded on said bad sector, and were recorded on this bad sector on said vacant sector, A control method of a memory given in claims 1 and 4 evacuating said hook field to said vacant sector.

[Claim 6]A control method of the memory according to claim 1, 2, 3, 4, or 5, wherein information which shows relation between each sector is recorded on said each field.

[Claim 7]A control method of the memory according to claim 1, wherein said memory is a storage which has restriction in writing frequencies.

[Claim 8]A control method of the memory according to claim 1, wherein said memory is a flash memory.

[Translation done.]

* NOTICES *

JPO and INPIT are not responsible for any damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.

2.**** shows the word which can not be translated.

3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention]This invention relates to the control method of a memory.

[0002]

[Description of the Prior Art]The memory storage for memorizing various application software other than RAM (random access memory) used in a program execution process is formed in information processors, such as a personal computer. As this kind of memory storage, since after power supply cutoff needed to hold that memory content, the hard disk drive using the magnetic recording disk as that storage was adopted.

[0003]However, instead of the above-mentioned magnetic recording disk, it succeeds in the trial using the non volatile semiconductor memory like a flash memory which can be written in with rapid-access-izing of the above-mentioned memory storage in recent years. By impressing high tension to the gate region of the memory cell, a flash memory makes an electric charge form in this gate region, and performs write-in memory of data.

[0004]However, in the flash memory, if the writing like the above concentrates on one memory cell, the characteristic degradation of the semiconductor itself arises and it has the problem that the memory cell will break.

[0005]

[Problem(s) to be Solved by the Invention]It succeeds in this invention considering this problem as a solution plug, and it is a thing.

The purpose is to provide the control method of the memory which can prolong a life.

[0006]

[Means for Solving the Problem]A control method of a memory by this invention classifies a record section of a memory with two or more sectors which each becomes from predetermined storage capacity, A fat entry which shows whether it is the method of controlling said memory by said sector unit, and each of said sector is a vacant sector is recorded all over a fat field of said memory, Write said file in a sector which searches a fat entry which shows that it is a vacant sector sequentially from a head of said fat field at the time of writing of a file, and is shown by this search fat entry, and. Rewrite for a fat entry which shows that it is not a vacant sector about said search fat entry, and at the time of elimination of a file. A file used as a candidate for deletion is deleted, and by deletion of said file, a fat entry about a sector used as a vacant sector is newly created, and this is recorded on the tail end of said fat field.

[0007]

[Embodiment of the Invention]Drawing 1 is a figure showing an example of the composition of an information processor. In drawing 1, the memory storage 10 comprises the memory controller 102 with which after power supply cutoff performs writing and read-out control to the flash memory 101 which can hold the memory content, and which can be written in, and this flash memory 101.

[0008]Drawing 2 is a figure showing the initial logical format of this flash memory 101. setting to the initial state, as it is shown in drawing 2 -- the storage area of the flash memory 101 -- a hook field, a directory region, and a FSL(Free Storage List)_FAT area -- and -- a file area -- it is classified into four fields. Access to this flash memory 101 is carried out in the sector unit shown by SID (sector ID)0-N with which each does not overlap, and the above-mentioned hook field is assigned to the sector of SID=0 at the time of the initial logical format.

[0009]In the following explanation, as for the flash memory 101, initialization or where data erasure is carried out, all the bits shall become logical-value "0." Although data erasure is possible only in a block unit, it enables the writing ("0" -> "1") to an elimination bit in word units (the number of predetermined bits). Drawing 3 is a figure showing the memory format at the time of the initial logical format of the above-mentioned hook field.

[0010]In drawing 3, the sector checkbite CB which shows whether discernment ID which shows that this one sector is a hook field, and the sector assigned to this hook field are normal is recorded on the head part of this hook field. D-SID which shows sector ID of the head of the above-mentioned directory region is recorded, and the postscript region is arranged after that. When above-mentioned D-SID is updated by the new thing, this postscript region is provided in order to add this newest D-SID one by one and to go.

[0011]In the gestalt of this example, sector ID in which a hook field exists shall be written in discernment ID, and the value which reversed all the bits of discernment ID shall be written in CB. Drawing 4 is a figure showing the memory format at the time of the initial logical format of the directory region shown in drawing 2. As it is shown in drawing 4, FSL directory entry 0-n and the directory entry 0 - k are formed in the directory region.

[0012]To the FSL directory entry 0 - each of n. Deletion checkbite DCB which shows whether this FSL directory entry is effective, All over a file type and a FSL_FAT area, FT-SID is recorded as sector ID which shows the sector on which the effective FAT entry (it mentions later) which exists in a very head position is recorded. after the time of an initial logical format and data are eliminated, all the bits of the FSL directory entry 0 - deletion checkbite DCB of each n are "logical-value "0 which shows that it is effective"." Namely, it becomes "effective" in the state where no data is written in, after elimination or initialization.

[0013]On the other hand, to the directory entry 0 - each of k. Deletion checkbite DCB which shows whether this directory entry is effective, floor line-SID which shows sector ID of the head on the file information (a file name, a file size, a file type, a recording date, etc.) about the file written in the file area of drawing 2 and the flash memory 101 on which this file is recorded is recorded. all the bits of the directory entry 0 - deletion checkbite DCB of each k are "logical-value "0 which shows that it is effective"" the time of an initial logical format, and after data erasure. Namely, initialization or after data erasure is carried out, it becomes "effective" in the state where no data is written in.

[0014]nSID which shows the relation information on the sector in this field is recorded on the head part of a directory region. Under the present circumstances, whenever all the storage capacity exceeds a part for one sector, as it is shown in drawing 4, nSID as sector relation information is recorded on that head part for a part for every one sector. With the file type in the FSL directory entry mentioned above and a directory entry. It is

the information which shows whether the file which a directory entry shows or the attribute of a subdirectory, and its entry are FSL directory entries, or it is a directory entry.

[0015] For example, if the point to which a directory entry points is a write-protected subdirectory, that attribute and the information which shows that this entry is a directory entry will be written in as a file type. Therefore, in explanation of drawing 4, although the inside of a directory region was clearly classified into the FSL directory entry and the directory entry, when actually using it, the kind of each entry can know the attribute from the fill type recorded in an entry. that is, it is not necessary to classify clearly a FSL directory entry and the field of each directory entry. In this case, even if the FSL directory entry and the directory entry are not clearly classified in the directory region, the relation place of each sector is understood by each nSID.

[0016] Drawing 5 is a figure showing the memory format at the time of the initial logical format of the FSL_FAT area shown in drawing 2. The fat pointer FP which shows each sector ID in the free regions (set of the sector which can be written in) which exist in the flash memory 101 to this FSL_FAT area, And the FAT entry which consists of a deletion checkbite DCB which shows whether the sector shown by this fat pointer FP is a vacant sector is formed. This FAT entry is provided corresponding to all the sectors in the free regions in this flash memory 101. Under the present circumstances, when all the bits of the above-mentioned deletion checkbite DCB are logical-value "0" which shows that it is a vacant sector, this FAT entry presupposes that it is an "effective" FAT entry. nSID which shows the relation information on the sector in this field is recorded on the head part of this FSL_FAT area. When the capacity of all the FAT entries exceeds the capacity for one sector, as it is shown in drawing 5, nSID as sector relation information is recorded on the head part for a part for every one sector. SID of the sector connected with the next of this sector is shown in this nSID.

[0017] Even if **** each field (a directory region, a FSL_FAT area, a file area) mentioned above is not arranged continuously, it can know the sector which should be connected with the next by nSID of each sector. That is, it can treat as a field which continued logically. In the file area of the flash memory 101 like the above, various application programs, an information file, etc. are memorized. The operating system (OS is called hereafter) program to which the control method of the memory by this invention was applied is memorized beforehand in the file area of the flash memory 101.

[0018] According to powering on, the above-mentioned OS program is read from this flash memory 101, and is written in the predetermined region of RAM(random access memory) 12. If the manual operating device 13 is operated so that a user may perform any one execution of the above-mentioned application program, this program will be read from the flash memory 101, and will be read into the above-mentioned RAM12. CPU(central processing unit) 11 executes the above-mentioned application program read on RAM12, and makes the executed result display on the display 14.

[0019] Under the present circumstances, by execution of this application software, if access of the writing of the file to the memory storage 10, elimination, etc. occurs, the memory controller 102 will perform control management which followed the above-mentioned OS to the flash memory 101. Just, although ** is shown in drawing 6, it explains to each operation of the writing of the new file in which it succeeds by execution of this OS program below, elimination, and updating taking the case of the flash memory 101 which has an initial logical format of ****.

[0020] Full capacity is 20 K bytes and the file area shown in drawing 6 comprises ten sectors which have the capacity of 2 K bytes respectively. Under the present circumstances, each sector is shown by sector ID:Q- (Q+9). sector ID:Q- (Q+9) corresponding to the sectors of each in the above-mentioned file area is recorded on the fat pointer FP of zero to FAT entry 9 each. the sector deletion checkbite DCB of zero to FAT entry 9 each is indicated to be by this fat pointer FP -- a vacant sector, i.e., writing, -- the logical value which shows that it is an effective sector -- it is "0." Under the present circumstances, the FAT entries 0-5 shall be recorded in the sector shown by sector ID:R, and the FAT entries 6-9 shall be recorded in the sector shown by sector ID: (R+1).

[0021] Therefore, sector ID:R of the sector in which the FAT entry 0 exists is recorded as FT-SID of the FSL directory entry 0. Deletion checkbite DCB of this FSL directory entry 0 is logical-value "0" which shows that this FSL directory entry 0 is effective. Five new file A-E in which each has the capacity which is 4 K bytes from a **** state although shown in this drawing 6 below is written in one by one, After eliminating this written-in file in order of the file C, E, and A, an example of operation until it writes in the file F which newly has the capacity of 6 K bytes is explained referring to drawing 7 - drawing 13.

[0022] Although shown in write-in drawing 6 of file A-E, in a **** state, the memory controller 102 reads first D-SID in the hook field shown in drawing 3. The memory controller 102 recognizes heading sector ID of a

directory region according to the contents of this read D-SID, and searches the effective (the logical value of DCB is "0") FSL directory entry which exists in a very head position in this sector. Under the present circumstances, as it is shown in drawing 6, the effective FSL directory entry which exists in a very head position is the FSL directory entry 0. The memory controller 102 searches the effective FAT entry (FAT entry whose logical value of DCB is "0") which exists in a very head position in the sector shown by sector ID:R currently recorded as FT-SID of this FSL directory entry 0. Under the present circumstances, the effective FAT entry which exists in a very head position in the sector shown by this sector ID:R is the FAT entry 0, as it is shown in drawing 6. Here, the memory controller 102 writes the file A in the sector shown by sector ID:Q currently recorded as the fat pointer FP of this FAT entry 0, as it is shown in drawing 7. Under the present circumstances, the capacity of the file A is size from the capacity for those with 4 K byte, and one sector (2 K bytes). Then, the memory controller 102 searches the effective FAT entry which exists in the next of the above-mentioned FAT entry 0. Under the present circumstances, the FAT entry searched is the FAT entry 1, as it is shown in drawing 6. Here, the memory controller 102 writes the continuation of the above-mentioned file A in the sector shown by sector ID: (Q+1) currently recorded as the fat pointer FP of this FAT entry 1, as shown in drawing 7.

[0023] Drawing 14 is a figure showing the format at the time of the writing of this file A. As it is shown in drawing 14, the first portion of the file A is written in the sector shown by sector ID:Q, and (Q+1) is recorded on the head position as nSID which shows the relation information on the following sector. this (Q+1) — the latter half of the file A is written in the sector shown. It leaves to "0", without writing anything in nSID which shows the relation information on the next sector of the head position. This shows that the relation to the following sector does not exist.

[0024] The new postscript to the file A becomes possible by writing in sector ID newly added to nSID of the sector of the last of the file A. Next, the memory controller 102 shows that the sectors of each shown by sector ID:Q and (Q+1) became write-in invalidity by rewriting the logical value of DCB of the above-mentioned FAT entries 0 and 1 to "1", as it is shown in drawing 7. Next, the memory controller 102 searches the head of a directory entry where it does not succeed in writing all over the above-mentioned directory region. Under the present circumstances, as indicated in drawing 7 as the directory entry of the head where it does not succeed in writing in the initial state like ****, it is the directory entry 0. Here the memory controller 102 to this directory entry 0. Sector ID:Q of the head on the file information (a file name, a file size, a file type, a recording date, etc.) about the above-mentioned file A and the flash memory 101 in which this file A is written is recorded.

[0025] By a series of operations like the above, the memory controller 102 completes the write operation of the file A. By the same method as the write operation of this file A, the memory controller 102 performs the writing of the file B and the file C one by one. drawing 8 is a figure showing the state of the flash memory 101 at the time of these files A and B and the writing of each C being completed.

[0026] As it is shown in drawing 8, the sectors of each shown by sector ID:Q- (Q+5) are used by the writing of file A-C. therefore, all the logical values of deletion checkbite DCB of zero to FAT entry 5 each which shows these sector ID:Q- (Q+5) are set to "1." Therefore, as it is shown in drawing 8, an effective FAT entry will not exist in the sector shown in sector ID:R. Under the present circumstances, as it exists in the very head position in a FSL_FAT area and an effective FAT entry is shown in drawing 8, it is the FAT entry 6, and sector ID of the sector on which this FAT entry 6 is recorded is (R+1). Then, the FSL directory entry which has not recorded the memory controller 102 into a FSL directory entry, and exists in the very head is searched. As shown in drawing 8, the FSL directory entry which exists in un-recording and the earliest head in a FSL directory entry, Since it is the FSL directory entry 1, the memory controller 102 records the above (R+1) as FT-SID of this FSL directory entry 1.

[0027] The DCB is set to "1" that the directory entry 0 which was "effective" until now should be made "invalidity." in the state by which it is shown in this drawing 8 — the next — in writing in the file D as a new file, the memory controller 102 reads first D-SID in the hook field shown in drawing 3. The memory controller 102 recognizes heading sector ID of a directory region according to the contents of this read D-SID, and searches the effective (the logical value of DCB is "0") FSL directory entry which exists in a very head position in this sector. Under the present circumstances, as it is shown in drawing 8, the effective FSL directory entry which exists in a very head position is the FSL directory entry 1. The memory controller 102 searches the effective (the logical value of DCB is "0") FAT entry which exists in a very head position in the sector shown without sector ID: (R+1) currently recorded as FT-SID of this FSL directory entry 1. Under the present circumstances,

the effective FAT entry which exists in a very head position in the sector shown by this sector ID: (R+1) is the FAT entry 6, as it is shown in drawing 8. Here, the memory controller 102 writes the file D in the sector shown by sector ID: (Q+6) currently recorded as the fat pointer FP of this FAT entry 6, as it is shown in drawing 9. Under the present circumstances, the capacity of the file D is size from the capacity for those with 4 K byte, and one sector (2 K bytes). Then, the memory controller 102 searches the effective FAT entry which exists in the next of the above-mentioned FAT entry 6. Under the present circumstances, the FAT entry searched is the FAT entry 7, as it is shown in drawing 8. Here, the memory controller 102 writes the continuation of the above-mentioned file D in the sector shown by sector ID: (Q+7) currently recorded as the fat pointer FP of this FAT entry 7, as it is shown in drawing 9.

[0028]Next, the memory controller 102 shows that the sectors of each shown by sector ID: (Q+6) and (Q+7) became write-in invalidity by rewriting the logical value of DCB of the above-mentioned FAT entries 6 and 7 to "1", as it is shown in drawing 9. Next, the memory controller 102 searches the head of a directory entry where it does not succeed in writing all over the above-mentioned directory region. Under the present circumstances, the directory entry of the head where it does not succeed in writing in the **** state although shown in drawing 8 is the directory entry 3. As it is shown in drawing 9, here the memory controller 102 to this directory entry 3. Sector ID: (Q+6) of the head on the file information (a file name, a file size, a file type, a recording date, etc.) about the above-mentioned file D and the flash memory 101 in which this file D is written is recorded.

[0029]By a series of operations like the above, the memory controller 102 completes the write operation of the file D. By the same method, the memory controller 102 writes in the file E. drawing 10 is a figure showing the state of the flash memory 101 at the time of the above-mentioned file A, B, and C, D, and the writing of each E being completed.

[0030]In eliminating the file C in the state by which it is shown in deletion of the file C, E, and A, or drawing 10 to cut, the memory controller 102 searches first the directory entry on which the file C is recorded as a file name out of the directory entry. Under the present circumstances, this directory entry is the directory entry 2, as it is shown in drawing 10. here, the memory controller 102 is recorded on sector ID: (Q+4) currently recorded as floor line-SID of this directory entry 2, and nSID of this sector -- the written contents of each sector shown in sector ID: (Q+5) which should be connected with the next are eliminated. The file C currently written in all over the file area is eliminated by this operation as it is shown in drawing 11.

[0031]Next, the memory controller 102 rewrites the logical value of DCB of the above-mentioned directory entry 2 to "1", as shown in drawing 11. Thereby, the contents of record of the above-mentioned directory entry 2 become invalid. Next, the memory controller 102 is written in all over the above-mentioned FSL_FAT area, and searches the tail end of the FAT entry of ending. Although shown in drawing 10, it writes in in a **** state and the tail end of the FAT entry of ending is the FAT entry 9. here, as the next FAT entry 10 of this FAT entry 9, and the fat pointer FP of 11 each, the memory controller 102 records (Q+4) and (Q+5), respectively, as it is shown in drawing 11. That is, (Q+4) and (Q+5) were created by the FAT entry as an effective FAT pointer in which these are shown by elimination of the file C since a file area (Q+4) and (Q+5) newly became free regions. here, no memory controllers 102 are written in DCB of these FAT entry 10 and 11 each.

[0032]By a series of operations like the above, the memory controller 102 completes the erasing operation of the file C. By the same method, the memory controller 102 eliminates the files E and A. Drawing 12 is a figure showing the state of the flash memory 101 at the time of elimination completing these files A, C, and E in order of C, E, and A.

[0033]In writing in the file F in which the capacity is 6 K bytes as a new file, in the state by which it is shown in the writing of the file F (6 K bytes), or drawing 12 to cut, the memory controller 102 reads D-SID in a hook field first. The memory controller 102 recognizes heading sector ID of a directory region according to the contents of this read D-SID, and searches the effective (the logical value of DCB is "0") FSL directory entry which exists in a very head position in this sector. Under the present circumstances, as it is shown in drawing 12, the effective FSL directory entry which exists in a very head position is the FSL directory entry 1. The memory controller 102 searches the effective (the logical value of DCB is "0") FAT entry which exists in a very head position in the sector shown by sector ID: (R+1) currently recorded as FT-SID of this FSL directory entry 1. Under the present circumstances, the effective FAT entry which exists in a very head position in the sector shown by this sector ID: (R+1) is the FAT entry 10, as it is shown in drawing 12. Here, the memory controller 102 writes the file F in the sector shown by sector ID: (Q+4) currently recorded as the fat pointer FP of this FAT entry 10, as it is shown in drawing 13. Under the present circumstances, the capacity of the file F is size from the capacity for those with 6 K byte, and one sector (2 K bytes). Then, the memory controller 102 searches the effective

FAT entry which exists in the next of the above-mentioned FAT entry 10. Under the present circumstances, the FAT entry searched is the FAT entry 11, as it is shown in drawing 12. Here, the memory controller 102 writes a continuation of the above-mentioned file F in the sector shown by sector ID: (Q+5) currently recorded as the fat pointer FP of this FAT entry 11, as it is shown in drawing 13. The memory controller 102 searches the effective FAT entry which exists in the next of the above-mentioned FAT entry 11 in order to write in a part for the remainder of the file F (2 K bytes). Under the present circumstances, the FAT entry searched is the FAT entry 12, as it is shown in drawing 12. Here, the memory controller 102 writes a part for the remainder of the above-mentioned file F (2 K bytes) in the sector shown by sector ID: (Q+8) currently recorded as the fat pointer FP of this FAT entry 12, as it is shown in drawing 13.

[0034]Drawing 15 is a figure showing the format at the time of the writing of this file F. As it is shown in drawing 15, a part of file F is written in the sector shown by sector ID: (Q+4), and (Q+5) is recorded on the head position as nSID which shows the relation information on the following sector. this (Q+5) — the continuation portion of the above-mentioned file F is written in, and (Q+8) is recorded on that head position as nSID which shows the relation information on the following sector by the sector shown. this (Q+8) — the final part of the above-mentioned file F is written in the sector shown. Since nothing is written in nSID which shows the relation information on the sector of the head position, it is still "0." This shows that the relation to the following sector does not exist as the file F.

[0035]next, as it is shown in drawing 13, the memory controller 102 to "1" the logical value of DCB of the FAT entries 10 and 11 about the above-mentioned file F, and 12 each, [rewrite and] It is shown that the sectors of each shown by sector ID: (Q+4), (Q+5), and (Q+8) became write-in invalidity. Next, the memory controller 102 searches the head of a directory entry where it does not succeed in writing all over the above-mentioned directory region. Under the present circumstances, the directory entry of the head where it does not succeed in writing in the **** state although shown in drawing 12 is the directory entry 5. As it is shown in drawing 13, here the memory controller 102 to this directory entry 5. Sector ID: (Q+4) of the head on the file information (a file name, a file size, a recording date, etc.) about the above-mentioned file F and the flash memory 101 in which this file F is written is recorded.

[0036]Here, as it is shown in drawing 13, the effective FAT entry does not exist in the sector shown by sector ID: R and (R+1) each. under the present circumstances -- it is under [FSL_FAT area] setting -- the head of an effective FAT entry -- sector ID -- : (R+2) -- it is the FAT entry 13 which exists in a sector. Then, the FSL directory entry which has not recorded the memory controller 102 into a FSL directory entry, and exists in the very head is searched. As shown in drawing 13, the FSL directory entry which exists in un-recording and the earliest head in a FSL directory entry is the FSL directory entry 2. Therefore, the memory controller 102 records the above (R+2) as FT-SID of this FSL directory entry 2.

[0037]Like the above, in the control method of the memory by this invention, as shown in drawing 6, the vacant sector (sector which can be written in) in a file area is realized to be linear list structure, and the pointer (fat pointer FP) of the linear list is recorded on a FSL_FAT area. Here, in the case of the writing of a file, as shown in drawing 7 - drawing 10, and drawing 13, a vacant sector is searched sequentially from the head position of the above-mentioned FSL_FAT area, it goes, and a sequential file is written in from this searched sector. Then, it rewrites to the deletion checkbite which shows that it is not logical-value "1", i.e., a vacant sector, about deletion checkbite DCB in the FAT entry which shows the sector in which this file was written. At the time of file deletion, as it is shown in drawing 11 and drawing 12, the file used as the candidate for deletion is deleted, and by this file deletion, the FAT entry about the sector used as a vacant sector is newly created, and this is added to the tail end of a FSL_FAT area, and it goes.

[0038]When securing free regions in order to write in data if memory management is performed as mentioned above, it will be used from the sector which saw serially and remains as free regions for a long time. Since only a specific sector is prevented from being used frequently at the time of renewal of a file according to this control method, it becomes possible to prolong the life of a flash memory.

[0039]he is trying to record the information (nSID) which shows the relation between each sector in a flash memory on the head position of each sector in the control method of the memory by this invention, as shown in drawing 3 - drawing 5, drawing 14, and drawing 15 Therefore, since the information which shows the relation between this sector will also be simultaneously eliminated at the time of deletion of a file if a sector is eliminated, as compared with the conventional memory control method which allotted the relation information on each sector to one place, it is avoidable that rewriting concentrates on a specific sector.

[0040]When all the directory entries in the heading sector of a directory region change into the state which

shows invalidity by writing, erasing operation, etc. of a **** file which were mentioned above, Or when the postscript region in this directory region becomes less than prescribed capacity, the memory controller 102 optimizes a directory region to the flash memory 101.

[0041]Optimization operation of the directory region by this memory controller 102 is explained below to optimization of a directory region, referring to drawing 13 and drawing 16. Under the present circumstances, drawing 16 (a) shows the recorded state before optimization of a directory region, and drawing 16 (b) shows an example of the recorded state after optimization, respectively. In an example shown in drawing 16 (a), the directory region before optimization shall comprise three sectors shown by sector ID:Y- (Y+2).

[0042]Here, in optimizing to the state of this drawing 16 (a), first, the memory controller 102 searches a vacant sector out of the free regions of the flash memory 101, and adds this vacant sector to the tail end sector as a directory region. For example, in the state of drawing 13, by referring to a FAT entry in order, the memory controller 102 judges with the sector shown by sector ID: (Q+9) being a vacant sector, and adds this sector as a tail end sector of a directory region, that is, it is shown in drawing 16 (b) at nSID of the sector shown in the state before optimization by sector ID: (Y+2) which was a tail end sector of the directory region -- as (Q+9) -- it describes. The memory controller 102 rewrites DCB of this FAT entry 13 to logical-value "1" that the FAT entry 13 shown in drawing 13 should be made invalid. By this operation, the sector shown by sector ID: (Q+9) serves as a directory region.

[0043]Next, all the effective directory entries which exist in the sector which is shown in the state of drawing 16 (a) by sector ID:Y which was a heading sector of the directory region as for the memory controller 102, As shown in drawing 16 (b), it copies to the sector shown by above-mentioned sector ID: (Q+9). Next, the memory controller 102 eliminates all the information currently recorded in the sector shown by this sector ID:Y, and as it is shown in drawing 16 (b), it makes free regions the sector shown by this sector ID:Y. That is, the memory controller 102 creates the FAT entry which should consider the sector of sector ID:Y as free-regions treatment, and adds this to the tail end of a FSL_FAT area.

[0044]By optimization of this directory region, the heading sector of a directory region turns into a sector shown by sector ID: (Y+1). Therefore, although the memory controller 102 is shown in drawing 3 considering sector ID: (Y+1) which shows the heading sector of this newest directory region as new D-SID, it adds a postscript and goes to the postscript region in a **** hook field.

[0045]If the postscript region in this hook field fills, the memory controller 102 will update a hook field to the flash memory 101. Under the present circumstances, as mentioned above, the hook field is arranged at the sector of sector ID:0 at the time of an initial format.

Updating drawing 17 of a hook field is a figure showing the subroutine flow for updating this hook field.

[0046]In drawing 17, the memory controller 102 memorizes first the newest D-SID recorded on the tail end of the hook field to the built-in register A (not shown) (Step S1). Next, the memory controller 102 eliminates all the contents of record in this hook field (Step S2). Next, the memory controller 102 rerecords the discernment ID and CB on the head of this hook field, respectively, and reads these (Step S3). next, the memory controller 102 judges whether it succeeded in these discernment ID and read-out of each CB normally (step S4). In this step S4, when judged with having succeeded in read-out normally, the memory controller 102 memorizes 9 to a built-in register (not shown) as the pointer (number of bytes) n in which a writing position is shown (Step S5). Next, the memory controller 102 records the newest D-SID memorized to the above-mentioned built-in register A on eye the n byte, and reads this to it (Step S6). That is, D-SID is recorded following the recorded discernment ID and CB. Next, the memory controller 102 judges whether it succeeded in read-out of this D-SID normally (Step S7). In this step S7, when judged with having succeeded in read-out normally, the memory controller 102 escapes from this hook field update routine, and returns to operation of a main routine (it does not explain).

[0047]On the other hand, in this step S7, when judged with not having succeeded in read-out normally, the memory controller 102 is memorized to a built-in register by using as the new pointer n what added 4 bytes to the above-mentioned pointer n (Step S8). Next, the memory controller 102 judges whether the pointer n is size from all the number-of-bytes N of hook area size, i.e., one sector, (step S9). In this step S9, when it judges that the pointer n is not size rather than all the number-of-bytes N of one sector, the memory controller 102 returns to operation of the above-mentioned step S6. On the other hand, in this step S9, when it judges that the pointer n is size rather than all the number-of-bytes N of one sector, the memory controller 102 eliminates all the contents of record of this sector (Step S10). after the end of this step S10, or in the above-mentioned step S4, when judged with not having succeeded in discernment ID and read-out of each CB normally, the memory controller 102 shifts to execution of a hook field evacuation routine (Step S11). That is, since it is thought in

this case that the sector currently assigned to the hook field is a bad sector, the memory controller 102 shifts to execution of the hook field evacuation routine like the following so that it may evacuate a hook field to another sector.

[0048]Drawing 18 is a figure showing this hook field evacuation routine. In drawing 18, first, the memory controller 102 searches for a vacant sector based on the contents of record of a FSL_FAT area, and elects it as a sector which should make this a new hook field (Step S22). Next, the memory controller 102 memorizes 1 to a built-in register (not shown) as the pointer n (Step S23). Next, the memory controller 102 records sector ID of this election sector on eye n byte of a bad sector, and reads this to it. The memory controller 102 records the data which carried out bit flipping of sector ID of this election sector on the byte (n+4) eye of this bad sector, and reads this to it (Step S24). Next, the memory controller 102 judges whether it succeeded in read-out by this step S24 normally (Step S25). In this step S25, when judged with not having succeeded in read-out normally, the memory controller 102 records data other than the bit-flipping data of sector ID of this election sector on the byte (n+4) eye of the above-mentioned bad sector, and reads this to it (Step S26). Next, the memory controller 102 judges whether it succeeded in read-out by this step S26 normally (Step S27). In this step S27, when judged with having succeeded in read-out normally, the memory controller 102 is memorized to a built-in register by using as the new pointer n what added 8 bytes to the above-mentioned pointer n (Step S28). Next, the memory controller 102 judges whether the pointer n is smallness from all the number-of-bytes N of one sector (Step S29). In this step S29, when it judges that the pointer n is smallness rather than all the number-of-bytes N of one sector, the memory controller 102 returns to operation of the above-mentioned step S24. When it judges that the pointer n is size rather than all the number-of-bytes N of one sector in this step S29 on the other hand, Or in the above-mentioned step S27, when judged with not having succeeded in read-out normally, the memory controller 102 notifies CPU11 shown in drawing 1 that shunting operation of the hook field went wrong (Step S30).

[0049]That is, if discernment ID in a hook field is sector ID in which this hook field exists and it is the value in which CB reversed all the bits of this sector ID, this hook field is normal and D-SID is recorded normally. For example, if discernment ID of the hook field in sector ID:0 is "00000000" (HEX) and CB is "FFFFFFFF" (HEX), this hook field is normal and D-SID is recorded on 4 bytes which continues after CB.

[0050]If discernment ID in a hook field is a value from which sector ID in which this hook field exists differs and it is the value in which CB reversed all the bits of this value, This hook field is a bad sector and having evacuated the contents of the hook field to the sector of sector ID which this discernment ID shows is shown. For example, if discernment ID of the hook field in sector ID:0 is "00000010" (HEX) and CB is "FFFFFFEF" (HEX), this sector is poor and having evacuated the contents of the hook field to sector ID:"10" (HEX) is shown.

[0051]If all the bits of the discernment ID and CB in a hook field cannot be respectively found in inversion relation mutually, it is poor and these 8 bytes of thing without a meaning is shown. Therefore, the following 8 bytes are investigated as the discernment ID and CB, and repeat execution of the **** inspection again mentioned above is carried out. If 8 bytes of field which is in inversion relation mutually (it is not poor) is not detected even if the sector of this hook field is completed, it means that evacuation of the sector had gone wrong.

[0052]When all the FAT entries in the renewal of a FSL_FAT area, in addition the heading sector of a FSL_FAT area change into the state which shows invalidity, the memory controller 102 updates a FSL_FAT area to the flash memory 101. For example, in the state of drawing 13, all the FAT entries 0-5 in the heading sector (sector ID:R) of a FSL_FAT area are in the state which shows invalidity. Therefore, the memory controller 102 eliminates all the information currently recorded on the sector shown by this sector ID:R, and makes this sector free regions. That is, the memory controller 102 creates the FAT entry which should consider the sector of sector ID:R as free-regions treatment, and adds this to the tail end of a FSL_FAT area.

[0053]When the elimination field in the final sector of a FSL_FAT area is completely lost, the memory controller 102 searches a vacant sector out of the free regions of the flash memory 101, and adds this vacant sector as a tail end sector of a FSL_FAT area. For example, in the state of drawing 13 the memory controller 102, By referring to the FAT entry in a FSL_FAT area in order, it judges with the sector shown by sector ID: (Q+9) being a vacant sector, and this sector is added as a tail end sector of a FSL_FAT area. That is, (Q+9) is described to nSID of the sector shown in the state before updating by sector ID: (R+2) which was a tail end sector of the FSL_FAT area. furthermore -- although the memory controller 102 is shown in drawing 13 -- as FT-SID of the **** FSL directory entry 3 -- a heading sector -- being shown (R+2) -- it records. Under the present

circumstances, DCB of the FSL directory entry 2 is rewritten to logical-value "1."

[0054] Since the number of erase times of a sector can be reduced according to the optimization operation to a hook field, a directory region, and the FSL_FAT areas of each like the above, validating the record section in the flash memory 101, it becomes possible to prolong the life of the flash memory 101. When optimizing each above-mentioned field, the elimination unit of the contents of record was used as one sector, but the effect same also as an elimination unit is acquired in the integral multiple of not only this but erase blocks.

[0055] Although the flash memory was used as a storage with the gestalt of this example, the invention in this application becomes effective to the memory which has restriction in writing frequencies in short.

[Translation done.]

* NOTICES *

JPO and INPIT are not responsible for any damages caused by the use of this translation.

1. This document has been translated by computer. So the translation may not reflect the original precisely.
2. **** shows the word which can not be translated.
3. In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] It is a figure showing an example of the composition of an information processor.

[Drawing 2] It is a figure showing the initial logical format of the flash memory 101.

[Drawing 3] It is a figure showing the initial format of a hook field.

[Drawing 4] It is a figure showing the initial format of a directory region.

[Drawing 5] It is a figure showing the initial format of a FSL_FAT area.

[Drawing 6] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 7] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 8] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 9] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 10] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 11] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 12] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 13] It is a figure showing an example of the state shift of the flash memory 101 by the control method of the memory of this invention.

[Drawing 14] It is a figure showing an example of the preservation gestalt of the file A.

[Drawing 15] It is a figure showing an example of the preservation gestalt of the file F.

[Drawing 16] It is a figure for explaining optimization operation of a directory region.

[Drawing 17] It is a figure showing the subroutine flow based on the updating operation of a hook field.

[Drawing 18] It is a figure showing the subroutine flow based on the saving operation of a hook field.

[Description of Notations in the Main Part]

10 Memory storage

101 Flash memory

102 Memory controller

[Translation done.]

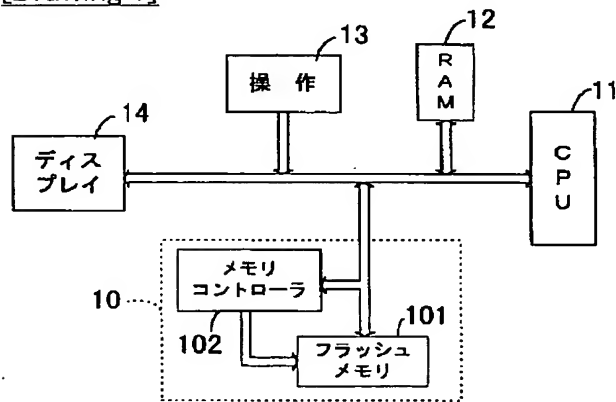
* NOTICES *

JP0 and INPIT are not responsible for any damages caused by the use of this translation.

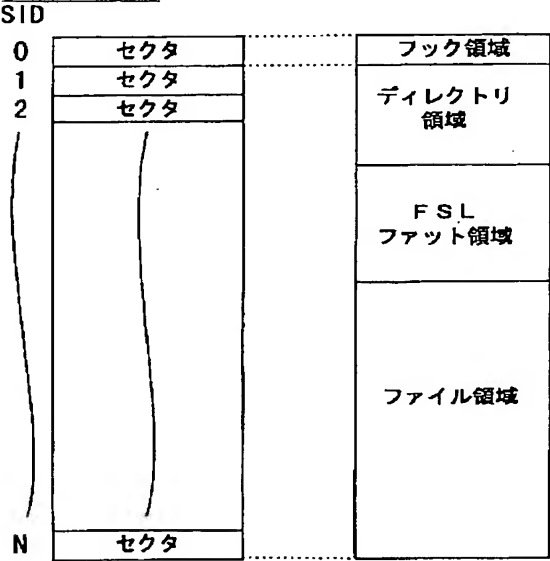
- 1.This document has been translated by computer. So the translation may not reflect the original precisely.
- 2.**** shows the word which can not be translated.
- 3.In the drawings, any words are not translated.

DRAWINGS

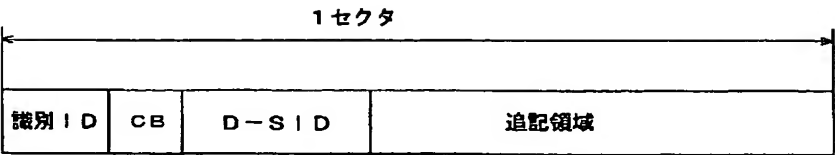
[Drawing 1]



[Drawing 2]



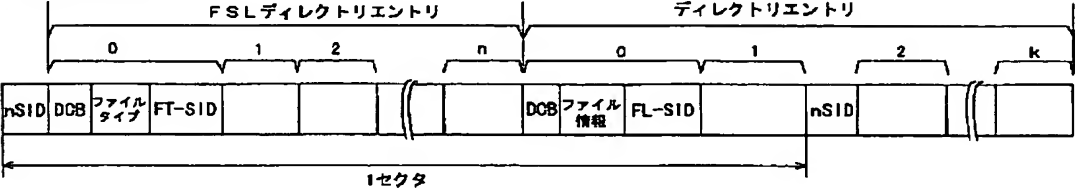
[Drawing 3]



フック領域
初期フォーマット

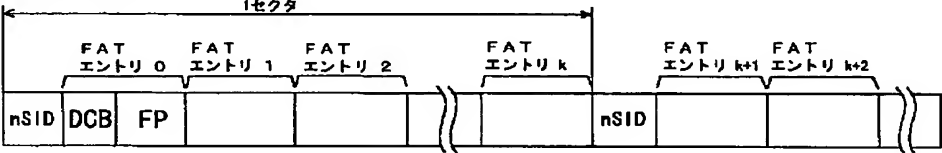
CB : セクタチェックバイト
D-SID : ディレクトリ領域先頭SID

[Drawing 4]



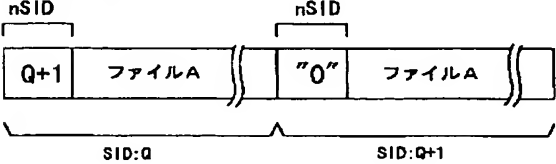
ディレクトリ領域
初期フォーマット

[Drawing 5]

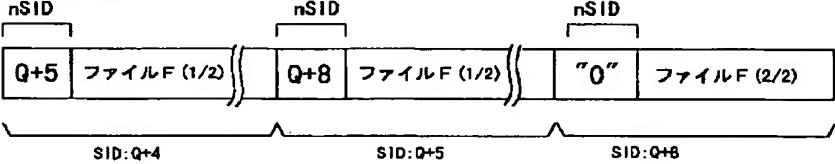


FSL_FAT領域
初期フォーマット

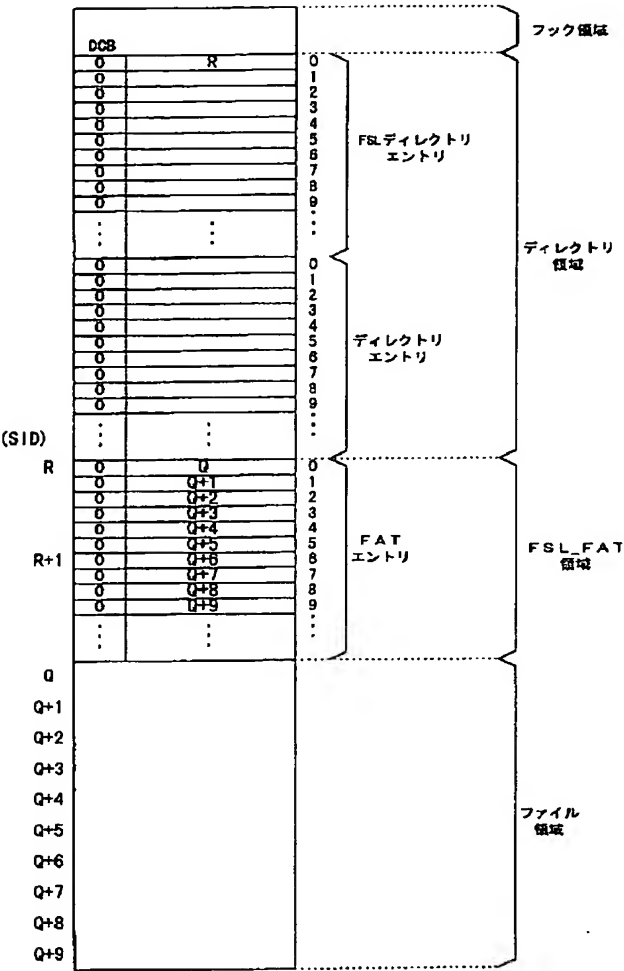
[Drawing 14]



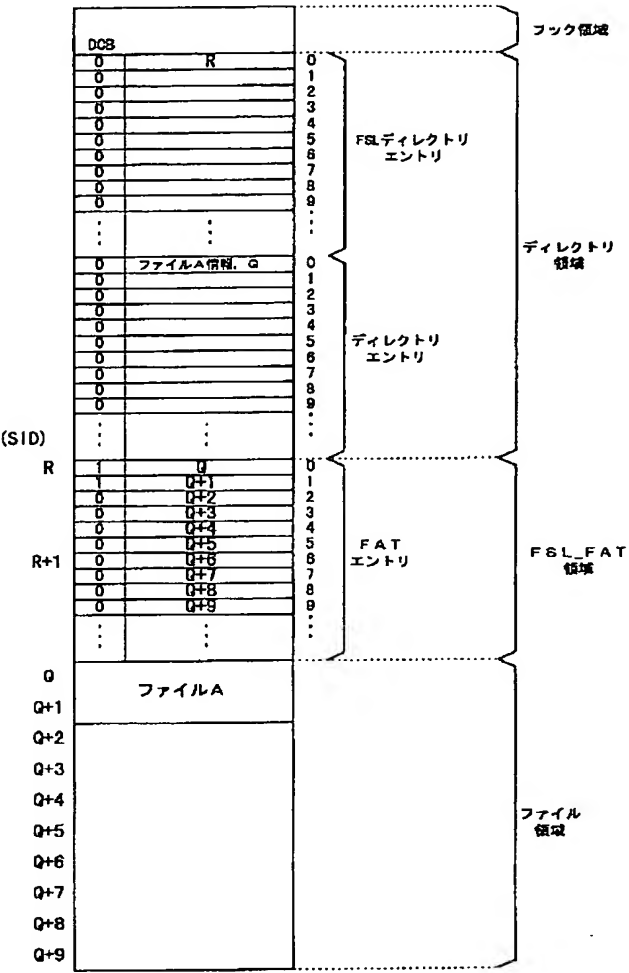
[Drawing 15]



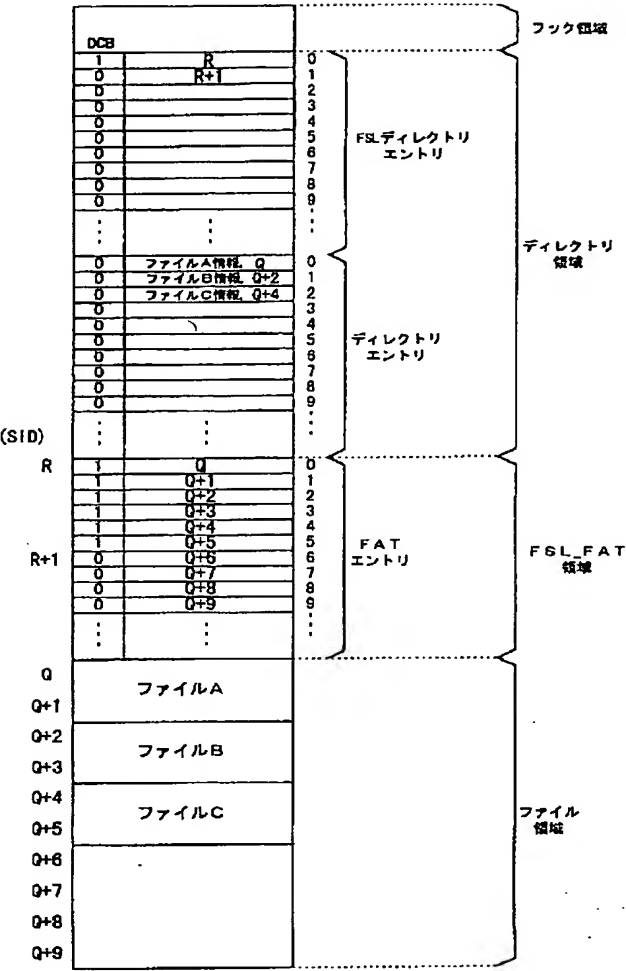
[Drawing 6]



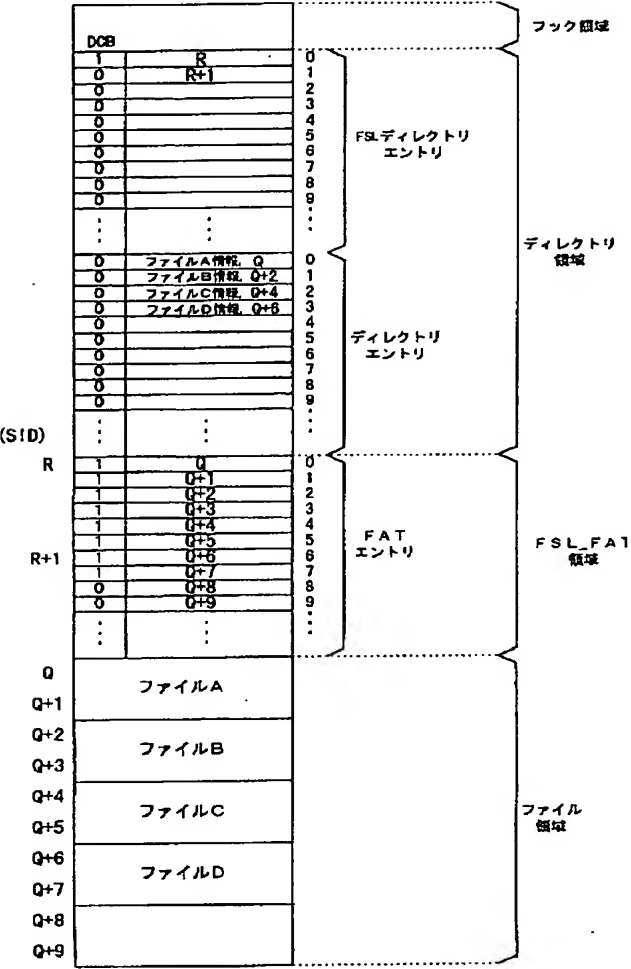
[Drawing 7]



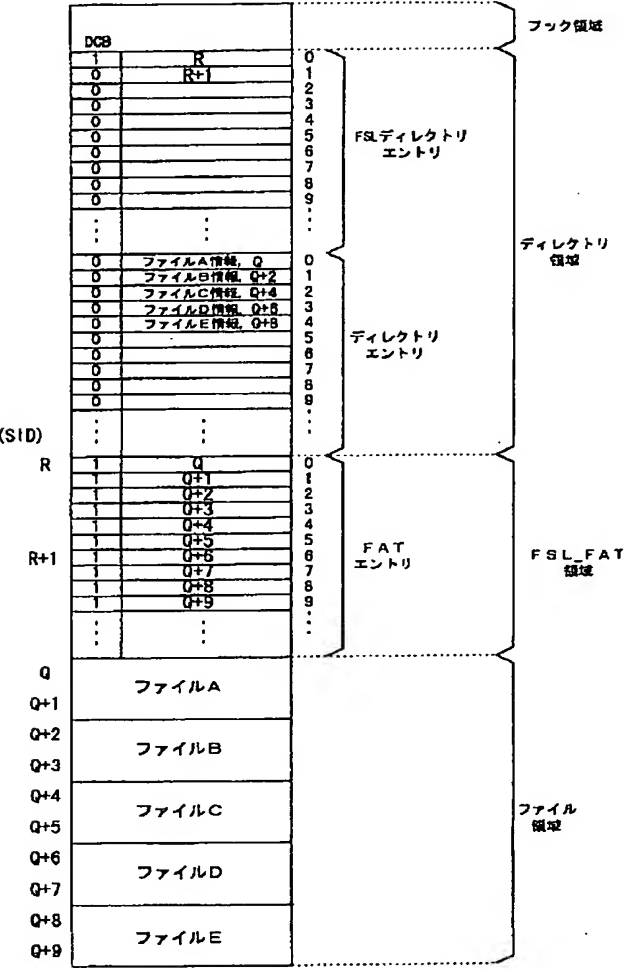
[Drawing 8]



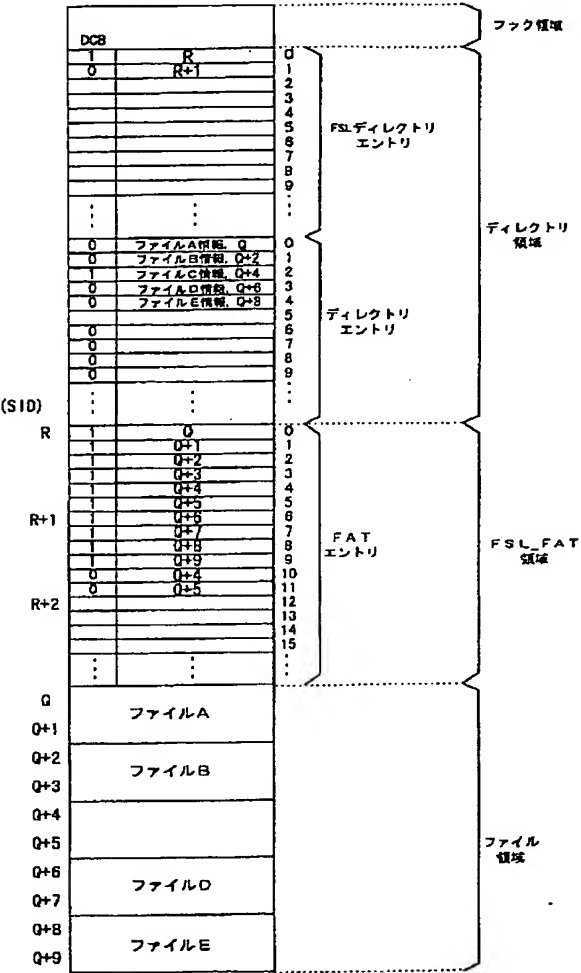
[Drawing 9]



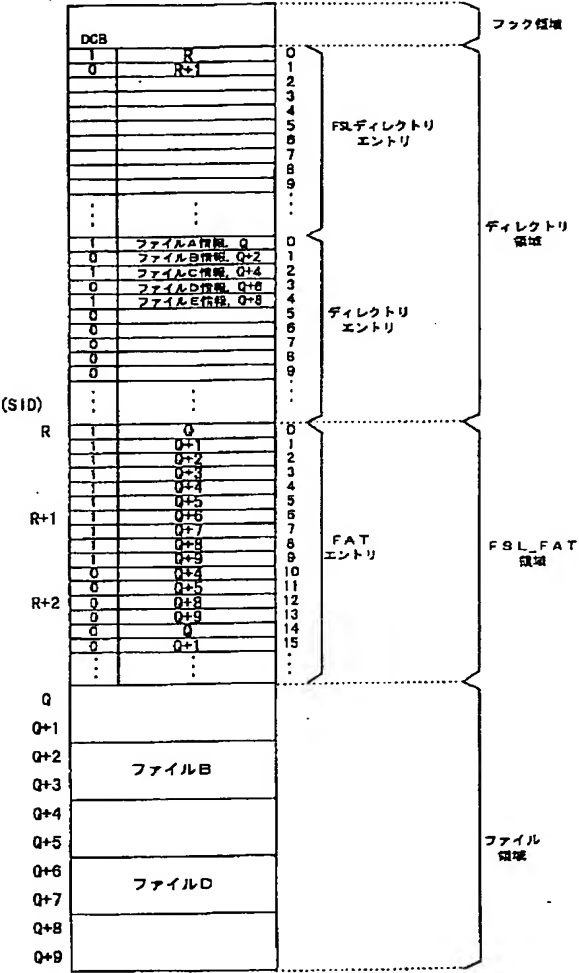
[Drawing 10]



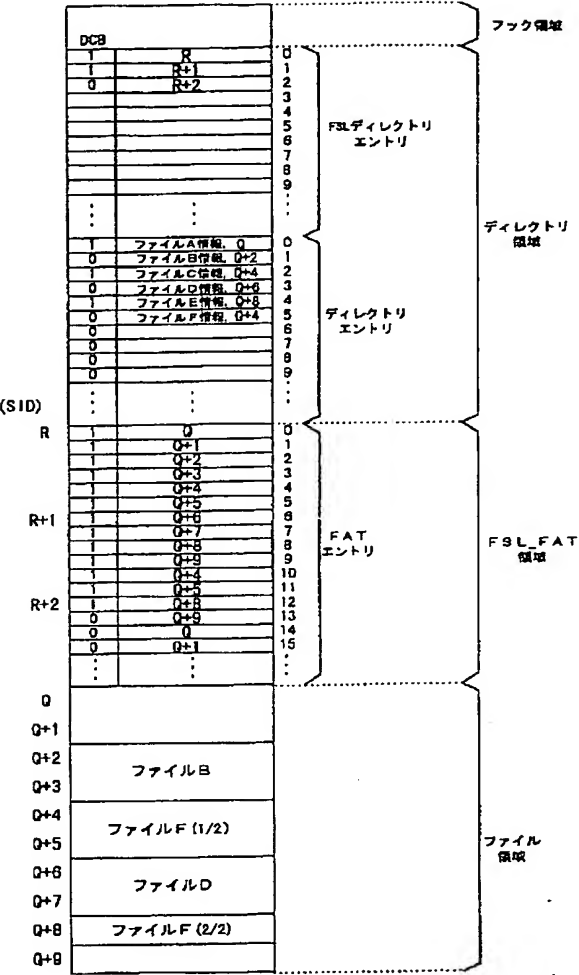
[Drawing 11]



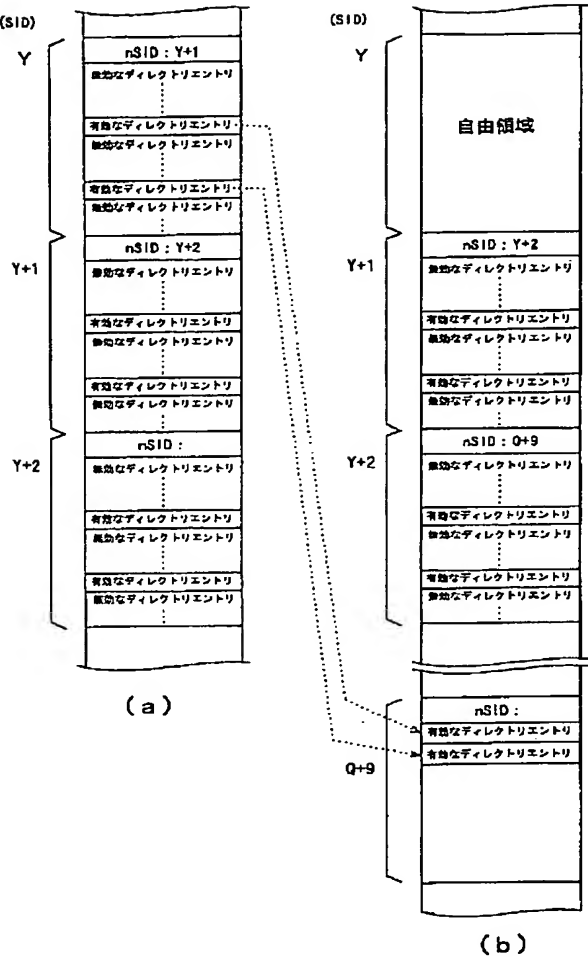
[Drawing 12]



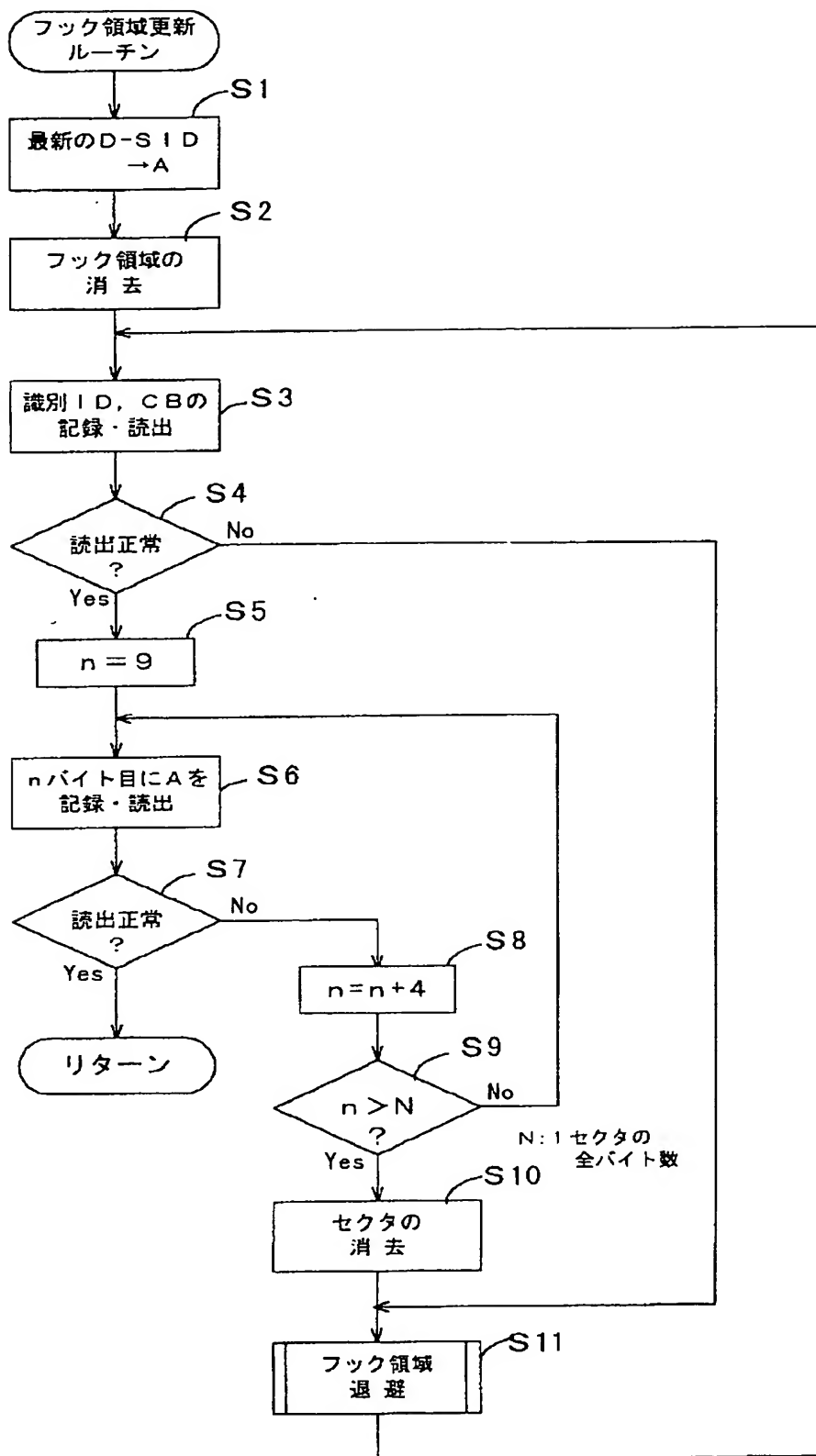
[Drawing 13]



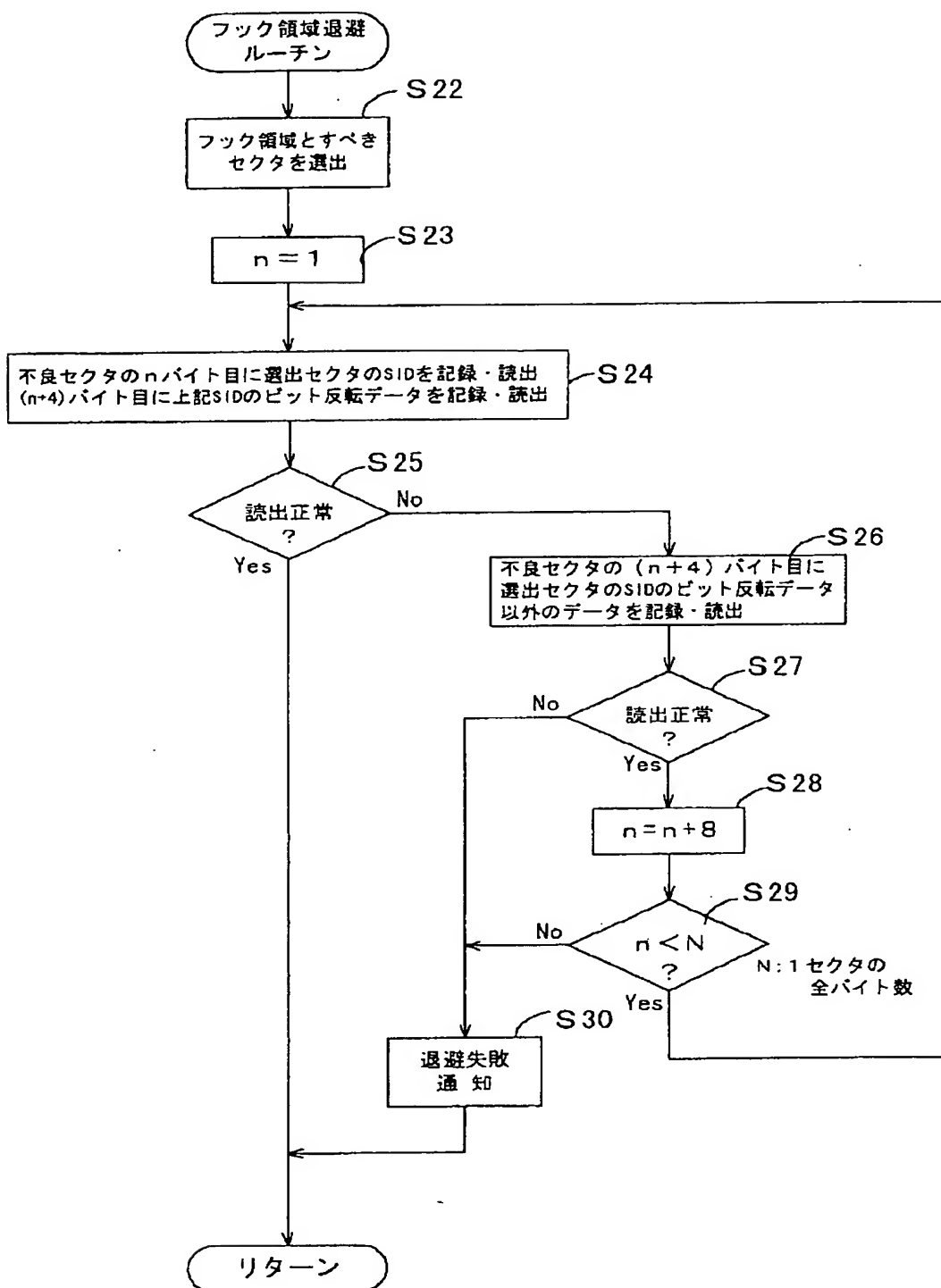
[Drawing 16]



[Drawing 17]



[Drawing 18]



[Translation done.]

例を示す図である。図1において、記憶装置10は、電源切断後もその記憶内容を持しておくことが出来る書き込み可能なフラッシュメモリ101、及びかかるフラッシュメモリ101に対して書き込み及び読出制御を行うメモリコントローラ102から構成される。

【0008】図2は、かかるフラッシュメモリ101の初期論理フォーマットを示す図である。図2に示されるが如く、その初期状態においてフラッシュメモリ101の記憶領域は、フック領域、ディレクトリ領域、FSL(Free Storage List)、FAT領域、及びファイル領域(セクタID)0～Nにて示されるセクタ単位に実施され、上記フック領域は、その初期論理フォーマット時においてSID=0のセクタに割り当てられる。

【0009】尚、以下の説明においてフラッシュメモリ101は、初期化またはデータ消去された状態では、全てのビットが論理値“0”になるものとする。又、データの消去は、ブロック単位でしかできないが、消去ビットの書き込み(“0”→“1”)は、ワード単位(所定ビット数)にて可能であるとする。図3は、上記フック領域の初期論理フォーマット時におけるメモリフォーマットを示す図である。

【0010】図3において、かかるフック領域の先頭部には、このセクタがフック領域であることを示す識別ID、及びこのフック領域に割り当てられたセクタが正常であるか否かを示すセクタチェックバイトCBが記録される。更に、上記ディレクトリ領域の先頭のセクタIDを示すD-SIDが記録され、その後、追記領域が配置されている。かかる追記領域は、上記D-SIDが新たなものに更新された場合に、この最新のD-SIDを順次追記して行く為に設けられている。

【0011】尚、本実施例の形態では、識別IDにはフック領域の存在するセクタIDを書き込み、CBには識別IDの全ビットを反転した値を書き込むものとする。図4は、図2に示されるディレクトリ領域の初期論理フォーマット時におけるメモリフォーマットを示す図である。図4に示されるが如く、ディレクトリ領域には、FSLディレクトリエントリ0～n、並びにディレクトリエントリ0～kが形成されている。

【0012】FSLディレクトリエントリ0～nの各々には、このFSLディレクトリエントリが有効なものであるか否かを示すディレクトリチェックバイトDCB、ファイルタイプ及びFSL_FAT領域中において最も先頭位置に存在する有効なFATエントリ(後述する)が記録されているセクタを示すセクタIDとして、FT-SIDが記録される。尚、初期論理フォーマット時、及びデータが消去された後においては、FSLディレクトリエントリ0～n各々のディレクトリチェックバイトDCBの全てのビットは、“有効”であることを示す論理値“0”と

なっている。すなわち、消去又は初期化後、データが何れも書き込まれていない状態の時に、“有効”となるのである。

【0013】一方、ディレクトリエントリ0～kの各々には、このディレクトリエントリが有効なものであるか否かを示すディレクトリチェックバイトDCB、図2のファイル領域に書き込まれたファイルに関するファイル情報(ファイル名、ファイルサイズ、ファイルタイプ、記録日時等)、並びに、このファイルが記録されているフラッシュメモリ101上の先頭のセクタIDを示すFSL-SIDが記録される。尚、初期論理フォーマット時、及びデータ消去後においては、ディレクトリエントリ0～k各々のディレクトリチェックバイトDCBの全てのビットは、“有効”であることを示す論理値“0”になっている。すなわち、初期化又はデータ消去された後、データが何れも書き込まれていない状態の時に“有効”となるのである。

【0014】又、ディレクトリ領域の先頭部には、この領域内でのセクタの繋がりが示すnSIDが記録される。この際、全ての記録容量が1セクタ分を超える度に、図4に示されるが如く、1セクタ分毎にその先頭部にセクタ繋がりが示すnSIDを記録する。尚、前述したFSLディレクトリエントリ及びディレクトリエントリ中のファイルタイプとは、ディレクトリエントリが示すファイル、あるいはサブディレクトリの属性、及びそのエントリがFSLディレクトリエントリであるかディレクトリエントリであることを示す情報である。

【0015】例えば、ディレクトリエントリによって指し示される先が書き込み禁止のサブディレクトリであれば、その属性、及びこのエントリがディレクトリエントリであることを示す情報がファイルタイプとして書き込まれることになる。従って、図4の説明では、ディレクトリ領域内を明確にFSLディレクトリエントリとディレクトリエントリとに区分けしたが、実際使用される場合には、夫々のエントリの種類はエントリ内に記録されるファイルタイプよりその属性を知ることができる。すなわち、FSLディレクトリエントリ、及びディレクトリエントリ各々の領域を明確に区分けする必要はない。この場合、ディレクトリ領域内でFSLディレクトリエントリとディレクトリエントリとが明確に区分けされていないくとも、各セクタの繋がりが先は夫々のnSIDにより分かる。

【0016】図5は、図2に示されるFSL_FAT領域の初期論理フォーマット時におけるメモリフォーマットを示す図である。かかるFSL_FAT領域は、フラッシュメモリ101に存在する自由領域(書き込み可能なセクタの集合)中における各セクタIDを示すファットポインタFP、及びこのファットポインタFPによって示されるセクタが空きセクタであるか否かを示すディレクトリチェックバイトDCBからなるFATエントリが形成

り0～9各々のディレクトリチェックバイトDCBは、このファットポインタFPによって示されるセクタが空きセクタ、すなわち有効なセクタであることを示す論理値“0”となっている。この際、FATエントリ0～5は、セクタID:Rにて示されるセクタ内に記録され、FATエントリ6～9は、セクタID:(R+1)にて示されるセクタ内に記録されるものとする。

【0021】従って、FSLディレクトリエントリ0のFT-SIDとして、FATエントリ0が存在するセクタのセクタID:Rが記録される。かかるFSLディレクトリエントリ0のディレクトリチェックバイトDCBは、このFSLディレクトリエントリ0が有効であることを示す論理値“0”となっている。以下に、かかる図6に示されるが如き状態から、各々が4Kバイトの容量を有する5つの新規ファイルA～Eを順次書き込み、この書き込んだファイルをファイルC、E、Aの順に消去した後、新たに6Kバイトの容量を有するファイルFを書き込むまでの動作例を、図7～図13を参照しつつ説明する。

【0022】ファイルA～Eの書込
図6に示されるが如き状態において、まず、メモリコントローラ102は、図3に示されるフック領域中のD-SIDを読出す。メモリコントローラ102は、この読み出したD-SIDの内容によってディレクトリ領域の先頭セクタIDを認識し、このセクタ内において最も先頭位置に存在する有効(DCBの論理値が“0”)なFSLディレクトリエントリを検索する。この際、図6に示されるが如く、最も先頭位置に存在する有効なFSLディレクトリエントリは、FSLディレクトリエントリ0である。メモリコントローラ102は、このFSLディレクトリエントリ0のFT-SIDとして記録されているセクタID:Rにて示されるセクタ内において、最も先頭位置に存在する有効なFATエントリ(DCBの論理値が“0”)であるFATエントリを検索する。この際、かかるセクタID:Rにて示されるセクタ内において、図6に示されるが如く、FATエントリ0である。ここ

で、メモリコントローラ102は、このFATエントリ0のファットポインタFPとして記録されているセクタID:Qにて示されるセクタに、図7に示されるが如く、ファイルAを書き込んで行く。この際、ファイルAの容量は4Kバイトあり、1セクタ分の容量(2Kバイト)よりも大である。そこで、メモリコントローラ102は、上記FATエントリ0の次に存在する有効なFATエントリを検索する。この際、検索されるFATエントリは、図6に示されるが如く、FATエントリ1である。ここで、メモリコントローラ102は、このFATエントリ1のファットポインタFPとして記録されているセクタID:(Q+1)にて示されるセクタに、図7に示されるように上記ファイルAの続きを書き込んで行くのである。

される。かかるFATエントリは、このフラッシュメモリ101における自由領域中の全てのセクタに対応して設けられる。この際、上記ディレクトリチェックバイトDCBの全てのビットが、空きセクタであることを示す論理値“0”である場合、このFATエントリは“有効”なFATエントリであるとする。又、かかるFSL_FAT領域の先頭部には、この領域内でのセクタの繋がりが示すnSIDが記録される。尚、全てのFATエントリの容量が1セクタ分の容量を超える場合には、図5に示されるが如く、1セクタ分毎にその先頭部にセクタ繋がりが示すnSIDを記録する。かかるnSIDに関する情報は、このセクタの次に繋がれるセクタのSIDが示されている。

【0017】尚、上述した如き各領域(ディレクトリ領域、FSL_FAT領域、ファイル領域)は連続して配列されていくなくても、各セクタのnSIDによって次に繋がるべきセクタを知ることが出来る。つまり、論理的には連続した領域として扱うことが出来るのである。上記の如きフラッシュメモリ101のファイル領域には、各種アプリケーションプログラム、及び情報ファイル等が記憶される。更に、フラッシュメモリ101のファイル領域には、予め、本発明によるメモリ制御方法が適用されたオペレーティングシステム(以下、OSと称する)プログラムが記憶されている。

【0018】電源投入に応じて上記OSプログラムは、かかるフラッシュメモリ101から読み出されてRAM(ランダムアクセスメモリ)12の所定領域に書き込まれる。使用者が上記アプリケーションプログラムのいずれか1つの実行を行うべく操作装置13の操作を行うと、このプログラムがフラッシュメモリ101から読み出されて上記RAM12に読み込まれる。CPU(中央処理装置)11は、RAM12上に読み込まれた上記アプリケーションプログラムを実行して、その実行結果をディスプレイ14上に表示せしめる。

【0019】この際、かかるアプリケーションソフトの実行により、記憶装置10に対するファイルの書込、及び消去等のアクセスが発生すると、メモリコントローラ102は、フラッシュメモリ101に対して上記OSに従った制御処理を実行する。以下に、かかるOSプログラムの実行行為とされる新規ファイルの書込、消去、及び更新の各動作について、図6に示されるが如き初期論理フォーマットを有するフラッシュメモリ101を例にとって説明する。

【0020】尚、図6に示されるファイル領域は、全容量が20Kバイトであり、各々2Kバイトの容量を有する10個のセクタから構成されている。この際、各セクタは、セクタID:Q～(Q+9)にて示される。又、FATエントリ0～9各々のファットポインタFPには、上記ファイル領域中のセクタ各々に対応したセクタID:Q～(Q+9)が記録される。又、FATエントリ

【0023】図14は、かかるファイルAの書き込み時のフォーマットを示す図である。図14に示されるが如く、セクタID:Qにて示されるセクタには、ファイルAの前半部分が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+1)が記録される。この(Q+1)にて示されるセクタには、ファイルAの後半部分が書き込まれる。その先頭位置の次のセクタの繋がり情報を示すnS1Dには何も書き込まず"0"のままにしておく。これは、次のセクタへの繋がり存在しないことを示すものである。

【0024】又、ファイルAへの新たな追加は、ファイルAの最後のセクタのnS1Dに新たに追加されたセクタIDを書き込むことにより可能となる。次に、メモリコントローラ102は、上記FATエントリ0及び1のDCBの論理値を、図7に示されるが如く"1"に書き換えることにより、セクタID:Q及び(Q+1)にて示されるセクタ各々が書き込まれたことを示す。次に、メモリコントローラ102は、上記ディレクトリ領域において書き込まれていないディレクトリエントリの先頭を検索する。この際、上述の如き初期状態において書き込みのされていない先頭のディレクトリエントリとは、図7に示されるように、ディレクトリエントリ0である。ここで、メモリコントローラ102は、かかるディレクトリエントリ0に、上記ファイルAに関するファイル情報(ファイル名、ファイルサイズ、ファイルタイプ、記録日時等)、並びに、このファイルAの書き込みが完了した際のフラッシュメモリ101上の先頭のセクタID:Qを記録する。

【0025】以上の如き一連の動作により、メモリコントローラ102は、ファイルAの書き込み動作を完了する。かかるファイルAの書き込み動作と同様な方法によって、メモリコントローラ102は、ファイルB、ファイルCの書き込みを行う。図8は、これらファイルA、B、及びC各々の書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0026】図8に示されるが如く、セクタID:Q〜(Q+5)によって示されるセクタ各々が、ファイルA〜Cの書き込みによって、これらセクタD:Q〜(Q+5)を示すFATエントリ0〜5各々のディレクティブDCBの論理値が全て"1"となる。よって、図8に示されるが如く、セクタID:Rに示されるセクタ内には、有効なFATエントリが存在しないこととなる。この際、FSL_FAT領域中の最も先頭位置に存在し、かつ有効なFATエントリは、図8に示されるが如くFATエントリ6であり、このFATエントリ6が記録されるセクタのセクタIDは(R+1)である。そこで、メモリコントローラ102は、FSLディレクトリエントリ中において、未記録であり、かつ最も先頭に存在するFSLディレクトリエントリを検索する。図8に示されるように、FSLディレクトリ

イレクトリエントリ3に、上記ファイルDに関するファイル情報(ファイル名、ファイルサイズ、ファイルタイプ、記録日時等)、並びに、このファイルDが書き込まれているフラッシュメモリ101上の先頭のセクタID:(Q+6)を記録する。

【0029】以上の如き一連の動作により、メモリコントローラ102は、ファイルDの書き込み動作を完了する。同様な方法によって、メモリコントローラ102は、ファイルEの書き込みを行う。図10は、上記ファイルA、B、C、D及びE各々の書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0030】ファイルC、E、Aの削除
かかる図10に示される状態においてファイルCの消去を行うにあたり、メモリコントローラ102は、先ず、ディレクトリエントリ中からファイル名としてファイルCが記録されているディレクトリエントリの検索を行う。この際、かかるディレクトリエントリは、図10に示されるが如く、ディレクトリエントリ2である。ここで、メモリコントローラ102は、かかるディレクトリエントリ2のFSL_S1Dとして記録されているセクタID:(Q+4)、及びこのセクタのnS1Dに記録されている次に繋がるべきセクタID:(Q+5)に示されるセクタ各々の書き込みを消去する。かかる動作により、ファイル領域に書き込まれていたファイルCは、図11に示されるが如く消去される。

【0031】次に、メモリコントローラ102は、上記ディレクトリエントリ2のDCBの論理値を、図11に示されるように、"1"に書き換える。これにより、上記ディレクトリエントリ2の記録内容が無効となる。次に、メモリコントローラ102は、上記FSL_FAT領域中において書き込み済みのFATエントリの最後尾を検索する。図10に示されるが如き状態において書き込み済みのFATエントリの最後尾は、FATエントリ9である。ここで、メモリコントローラ102は、かかるFATエントリ9の次のFATエントリ10及び11各々のDCBに何も書き込まない。

【0032】以上の如き一連の動作により、メモリコントローラ102は、ファイルCの消去動作を完了する。同様な方法によって、メモリコントローラ102は、ファイルA、C、及びEの消去を行う。図12は、これらファイルA、C、及びEの消去が完了した際のフラッシュメモリ101の状態を示す図である。

【0033】ファイルF(6Kバイト)の書き込み

かかる図12に示される状態において、新規ファイルとして、その容量が6KバイトのファイルFの書き込みを行うにあたり、先ず、メモリコントローラ102は、FAT領域中のD〜S1Dを読み出す。メモリコントローラ102は、この読み出したD〜S1Dの内容をよってディレクティブ領域に先頭セクタIDを認識し、このセクタ内において最も先頭位置に存在する有効(DCBの論理値が"0")なFSLディレクトリエントリを検索する。この際、図12に示されるが如く、最も先頭位置に存在する有効なFSLディレクトリエントリは、FSLディレクトリエントリ1である。メモリコントローラ102は、このFSLディレクトリエントリ1のFAT_S1Dとして記録されているセクタID:(R+1)にて示されるセクタ内において、最も先頭位置に存在する有効(DCBの論理値が"0")なFATエントリを検索する。この際、かかるセクタID:(R+1)にて示されるセクタ内において、最も先頭位置に存在する有効なFATエントリは、図12に示されるが如く、FATエントリ10である。ここで、メモリコントローラ102は、このFATエントリ10のファットポイントPとして記録されているセクタID:(Q+4)にて示されるセクタに、図13に示されるが如く、ファイルFの書き込みを行き込んで行く。この際、ファイルFの容量は6Kバイトあり、1セクタ分の容量(2Kバイト)よりも大である。そこで、メモリコントローラ102は、上記FATエントリ10の次に存在する有効なFATエントリを検索する。この際、検索される有効なFATエントリは、図12に示されるが如く、FATエントリ11である。こ

で、メモリコントローラ102は、このFATエントリ11のファットポイントPとして記録されているセクタID:(Q+5)にて示されるセクタに、図13に示されるが如く、上記ファイルFの続きを書き込んで行く。更に、ファイルFの残り分(2Kバイト)を書き込むべく、メモリコントローラ102は、上記FATエントリ11の次に存在する有効なFATエントリの検索を実施する。この際、検索されるFATエントリは、図12に示されるが如く、FATエントリ12である。ここで、メモリコントローラ102は、このFATエントリ12のファットポイントPとして記録されているセクタID:(Q+8)にて示されるセクタに、図13に示されるが如く上記ファイルFの残り分(2Kバイト)を書き込む。

【0034】図15は、かかるファイルFの書き込み時のフォーマットを示す図である。図15に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの続き部分が書き込まれ、共に、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0035】図16は、かかるファイルFの書き込み時のフォーマットを示す図である。図16に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0036】図17は、かかるファイルFの書き込み時のフォーマットを示す図である。図17に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0037】図18は、かかるファイルFの書き込み時のフォーマットを示す図である。図18に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0038】図19は、かかるファイルFの書き込み時のフォーマットを示す図である。図19に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0039】図20は、かかるファイルFの書き込み時のフォーマットを示す図である。図20に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0040】図21は、かかるファイルFの書き込み時のフォーマットを示す図である。図21に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0041】図22は、かかるファイルFの書き込み時のフォーマットを示す図である。図22に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0042】図23は、かかるファイルFの書き込み時のフォーマットを示す図である。図23に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0043】図24は、かかるファイルFの書き込み時のフォーマットを示す図である。図24に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0044】図25は、かかるファイルFの書き込み時のフォーマットを示す図である。図25に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0045】図26は、かかるファイルFの書き込み時のフォーマットを示す図である。図26に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0046】図27は、かかるファイルFの書き込み時のフォーマットを示す図である。図27に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0047】図28は、かかるファイルFの書き込み時のフォーマットを示す図である。図28に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

【0048】図29は、かかるファイルFの書き込み時のフォーマットを示す図である。図29に示されるが如く、セクタID:(Q+4)にて示されるセクタには、ファイルFの一部が書き込まれ、その先頭位置には、次のセクタの繋がり情報を示すnS1Dとして(Q+5)が記録される。この(Q+5)にて示されるセクタには、上記ファイルFの書き込みが完了した際のフラッシュメモリ101の状態を示す図である。

ると共に、このファイル削除によって空きセクタとなつたセクタに関するFATエントリを新たに作成し、これをFSL_FAT領域の最後尾に追記して行く。

【0038】以上のようにメモリ管理を行えば、データを書き込むべく自由領域を確保する際に、時系列的に見て最も長く自由領域として残っているセクタから使用することになる。かかる制御方法によれば、ファイルの更新時において、特定のセクタだけが頻繁に使用されることが防止されるので、フラッシュメモリの寿命を延ばすことが可能となるのである。

【0039】又、本発明によるメモリの制御方法においては、図3～図5、図14及び図15に示されるように、フラッシュメモリ内の各セクタ間の繋がりを示す情報(nSID)をセクタ毎の先頭位置に記録するようになっている。よって、ファイルの削除時には、セクタの消去を行えばこのセクタ間の繋がりを示す情報も同時に消去されるので、各セクタの繋がりが一か所に配した従来のメモリ管理方法に比して、特定のセクタに書き換えが集中することを避けることが出来るのである。

【0040】尚、前述した如きファイルの書込及び消去操作等により、ディレクトリ領域の先頭セクタ内における全てのディレクトリエントリが無効を示す状態となった場合、あるいは、このディレクトリ領域中の追記領域が所定容量よりも少なくなった場合、メモリコントロール102は、フラッシュメモリ101に対してディレクトリ領域の最適化を行う。

【0041】ディレクトリ領域の最適化

以下に、かかるメモリコントロール102によるディレクトリ領域の最適化動作について、図13及び図16を参照しつつ説明する。この際、図16(a)は、ディレクトリ領域の最適化前の記録状態、図16(b)は、最適化後の記録状態の一例を示すものである。又、図16(a)に示される一例においては、最適化前のディレクトリ領域はセクタID:Y～(Y+2)にて示される3つのセクタから構成されているものとする。

【0042】ここで、かかる図16(a)の状態に対して最適化を実施するにあたり、メモリコントロール102は、まず、フラッシュメモリ101の自由領域中から空きセクタを検索し、この空きセクタをディレクトリ領域としての最後尾セクタに追加する。例えば、図13の状態においては、メモリコントロール102は、FATエントリを順に参照することにより、セクタID:(Q+9)にて示されるセクタが空きセクタであると判定し、このセクタをディレクトリ領域の最後尾セクタとして追加する。すなわち、最適化前の状態においてディレクトリ領域の最後尾セクタであったセクタID:(Y+2)にて示されるセクタのnSIDに、図16(b)に示される如く(Q+9)を記録するのである。更に、メモリコントロール102は、図13に示されるFATエントリ13を無効とすべく、かかるFATエントリ13のD

CBを論理値"1"に書き換える。かかる動作により、セクタID:(Q+9)にて示されるセクタは、ディレクトリ領域となる。

【0043】次に、メモリコントロール102は、図16(a)の状態においてディレクトリ領域の先頭セクタであったセクタID:Yにて示されるセクタ内に存在する全ての有効なディレクトリエントリを、図16(b)に示されるように、上記セクタID:(Q+9)にて示されるセクタにコピーする。次に、メモリコントロール102は、かかるセクタID:Yにて示されるセクタ内に記録されていた情報を全て消去して、図16(b)に示される如く、このセクタID:Yにて示されるセクタを自由領域とする。すなわち、メモリコントロール102は、セクタID:Yのセクタを自由領域扱いとすべくFATエントリを作成し、これをFSL_FAT領域の最後尾に追記するのである。

【0044】かかるディレクトリ領域の最適化により、ディレクトリ領域の先頭セクタは、セクタID:(Y+1)にて示されるセクタとなる。従って、メモリコントロール102は、この最新のディレクトリ領域の先頭セクタを示すセクタID:(Y+1)を、新たなD-SIDとして図3に示される如きフック領域中の追記領域に追記して行く。

【0045】尚、かかるフック領域中の追記領域が一杯になると、メモリコントロール102は、フラッシュメモリ101に対してフック領域の更新を行う。この際、上述した如く初期フォーマット時においてフック領域は、セクタID:0のセクタに配置されている。フック領域の更新

図17は、かかるフック領域の更新を実施する為のサブルーチンフローを示す図である。

【0046】図17において、メモリコントロール102は、まず、フック領域の最後尾に記録された最新のD-SIDをその内蔵レジスタA(図示せず)に記憶する(ステップS1)。次に、メモリコントロール102は、かかるフック領域の全ての記録内容を消去する(ステップS2)。次に、メモリコントロール102は、識別ID及びCBをかかるフック領域の先頭に夫々記録し直してこれらを読み出す(ステップS3)。次に、メモリコントロール102は、これら識別ID及びCB各々の読み出しが正常に為されたか否かの判定を行う(ステップS4)。かかるステップS4において、読み出しが正常に為されたと判定された場合、メモリコントロール102は、書き込み位置を示すポインタ(バイト数)nとして9を内蔵レジスタA(図示せず)に記憶する(ステップS5)。次に、メモリコントロール102は、そのnバイト目に、上記内蔵レジスタAに記憶されている最新のD-SIDを記録し、これを読み出す(ステップS6)。すなわち、記録された識別ID及びCBに続いてD-SIDを記録するのである。次に、メモリ

コントロール102は、かかるD-SIDの読み出しが正常に為されたか否かの判定を行う(ステップS7)。かかるステップS7において、読み出しが正常に為されたと判定された場合、メモリコントロール102は、このフック領域更新ルーチンを抜けてメインルーチン(図明せず)の動作に戻る。

【0047】一方、かかるステップS7において、読み出しが正常に為されなかったと判定された場合、メモリコントロール102は、上記ポインタnに4バイトを加算したものを新たなポインタnとし内蔵レジスタに記憶する(ステップS8)。次に、メモリコントロール102は、そのポインタnがフック領域の大きさ、すなわち1セクタの全バイト数Nよりも大であるか否かの判定を行う(ステップS9)。かかるステップS9において、ポインタnが1セクタの全バイト数Nよりも大ではないと判定された場合、メモリコントロール102は、上記ステップS6の動作に戻る。一方、かかるステップS9において、ポインタnが1セクタの全バイト数Nよりも大であると判定された場合、メモリコントロール102は、このセクタの記録内容を全て消去する(ステップS10)。かかるステップS10の終了後、あるいは、上記ステップS4において、識別ID及びCB各々の読み出しが正常に為されなかったと判定された場合、メモリコントロール102は、フック領域更新ルーチンの実行に移る(ステップS11)。つまり、この際、フック領域が割り当てられていたセクタが不良セクタであると考えられるので、メモリコントロール102は、別のセクタにフック領域を遷移させるべく以下の如きフック領域更新ルーチンの実行に移るのである。

【0048】図18は、かかるフック領域更新ルーチンを示す図である。図18において、まず、メモリコントロール102は、FSL_FAT領域の記録内容に基いて空きセクタを探索し、これを新たなフック領域とすべきセクタとして選出する(ステップS2)。次に、メモリコントロール102は、ポインタnとして1を内蔵レジスタ(図示せず)に記憶する(ステップS23)。次に、メモリコントロール102は、不良セクタのnバイト目に、この選出セクタのセクタIDを記録し、これを読み出す。更に、メモリコントロール102は、この不良セクタの(n+4)バイト目に、この選出セクタのセクタIDをビット反転したデータを記録し、これを読み出す(ステップS24)。次に、メモリコントロール102は、かかるステップS24による読み出しが正常に為されたか否かの判定を行う(ステップS25)。かかるステップS25において、読み出しが正常に為されなかったと判定された場合、メモリコントロール102は、上記不良セクタの(n+4)バイト目に、この選出セクタのセクタIDのビット反転データ以外のデータを記録し、これを読み出す(ステップS26)。次に、メモリコントロール102は、かかるステップS26によ

る読み出しが正常に為されたか否かの判定を行う（ステップS27）。かかるステップS27において、読み出しが正常に為されたと判定された場合、メモリコントローラ102は、上記ポイントnに8バイトを加算したものを新たなポイントnとして内蔵レジスタに記憶する（ステップS28）。次に、メモリコントローラ102は、そのポイントnが1セクタの全バイト数Nよりも小さいステップS29において、ポイントnが1セクタの全バイト数Nよりも大であると判定された場合、メモリコントローラ102は、上記ステップS27において、読み出しが正常に為されなかったと判定された場合、メモリコントローラ102は、フック領域の待避動作が失敗した旨を、図1に示されるCPU11に通知する（ステップS30）。

【0049】すなわち、フック領域中の識別IDがそのフック領域が存在するセクタIDであり、かつCBがそのセクタIDの全ビットを反転した値であれば、このフック領域は正常でありD-SIDが正常に記録されている。例えば、セクタID:0にあるフック領域の識別IDが"00000000"(HEX)であり、かつCBが"FFFFFF"(HEX)であれば、このフック領域は正常であり、CBの後には続く4バイトはD-SIDが記録されている。

【0050】また、フック領域中の識別IDがそのフック領域が存在するセクタIDとは異なる値であり、かつCBがその値の全ビットを反転した値であれば、このフック領域は不良セクタであり、この識別IDが示すセクタIDのセクタへフック領域の内容を退避してあることを示し、例えば、セクタID:0にあるフック領域の識別IDが"00000010"(HEX)であり、かつCBが"FFFFFF"(HEX)であれば、このセクタは不良であり、フック領域の内容はセクタID:"10"(HEX)へ退避してあることを示す。

【0051】更に、フック領域中の識別IDとCBの全ビットが互いに反転関係になれば、この8バイトは不良であり意味を持たないことを示す。従って、次の8バイトを識別ID及びCBとして調べ、再び上述した如き検査を繰り返して実行する。もし、かかるフック領域のセクタが繰り足しても互いに反転関係にある（不良ではない）8バイトの領域が検出されなければセクタの退避が失敗したこととなるのである。

【0052】FSL_FAT領域の更新

尚、FSL_FAT領域の先頭セクタ内における全てのFATエントリが無効を示す状態となった場合、メモリコントローラ102は、フラッシュメモリ101に対してFSL_FAT領域の更新を行う。例えば、図13の状態においては、FSL_FAT領域の先頭セクタ（セ

クタID:R）内の全てのFATエントリ0～5が無効を示す状態となっている。よって、メモリコントローラ102は、このセクタID:Rにて示されるセクタに記録されている情報を全て消去し、かかるセクタを自由領域とする。すなわち、メモリコントローラ102は、セクタID:Rのセクタを自由領域扱いとすべきFATエントリを作成し、これをFSL_FAT領域の最後尾に追記するのである。

【0053】尚、FSL_FAT領域の最終セクタ内における消去領域が全く無くなった場合、メモリコントローラ102は、フラッシュメモリ101の自由領域から空きセクタを検出し、この空きセクタをFSL_FAT領域の最後尾セクタとして追加する。例えば、図13の状態においては、メモリコントローラ102は、FSL_FAT領域中のFATエントリを順に参照することにより、セクタID:(Q+9)にて示されるセクタが空きセクタであると判定し、このセクタをFSL_FAT領域の最後尾セクタとして追加する。すなわち、更新前の状態においてFSL_FAT領域の最後尾セクタであったセクタID:(R+2)にて示されるセクタのn SIDに(Q+9)を記述するのである。更に、メモリコントローラ102は、図13に示されるが如きFSL_FATエントリ3のFT-SIDとして、先頭セクタを示す(R+2)を記録するのである。この際、FSL_FATエントリ2のDCBを論理値"1"に書き換える。

【0054】以上の如き、フック領域、ディレトリ領域、及びFSL_FAT領域各々に対する最適化動作によれば、フラッシュメモリ101内の記録領域を有効化しつつセクタの消去回数を減らすことが出来るので、フラッシュメモリ101の寿命を延ばすことが可能となる。尚、上記各領域を最適化する際、記録内容の消去単位を1セクタとしたが、これに限らず消去ブロックの整数倍を消去単位としても同様の効果が得られる。

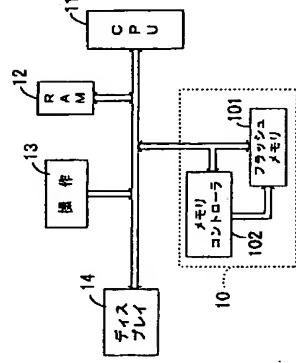
【0055】又、本実施例の形態ではフラッシュメモリを記憶媒体として用いたが、要するに書き込み回数に制限のあるメモリに対して本願発明は有効となるのである。

【図面の簡単な説明】

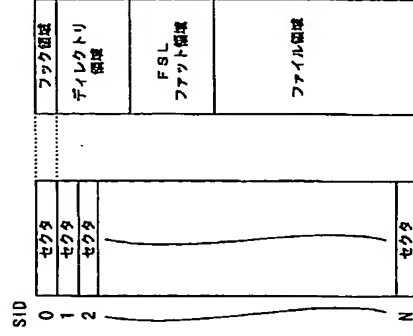
- 【図1】情報処理装置の構成の一例を示す図である。
【図2】フラッシュメモリ101の初期論理フォーマットを示す図である。
【図3】フック領域の初期フォーマットを示す図である。
【図4】ディレトリ領域の初期フォーマットを示す図である。
【図5】FSL_FAT領域の初期フォーマットを示す図である。
【図6】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。

- 【図7】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。
【図8】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。
【図9】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。
【図10】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。
【図11】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。
【図12】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。
【図13】本発明のメモリの制御方法によるフラッシュメモリ101の状態推移の一例を示す図である。

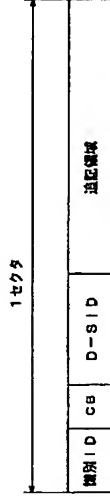
【図1】



【図2】



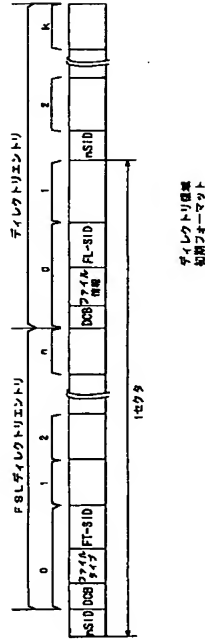
【図3】



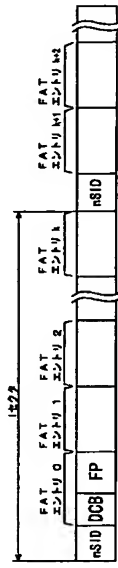
フック領域
初期フォーマット

CB: セクタチェックバイト
D-SID: ディレトリ領域先頭SID

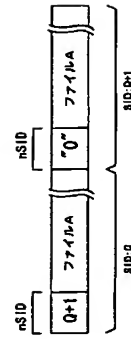
【図4】



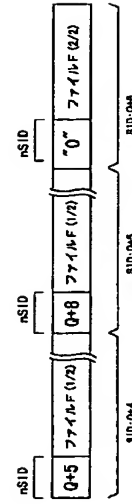
【图5】



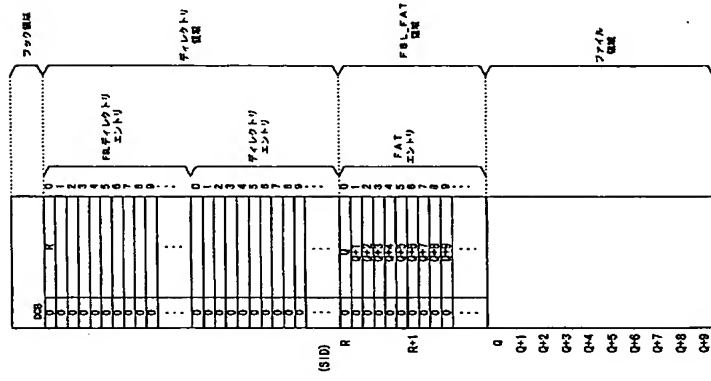
【图14】



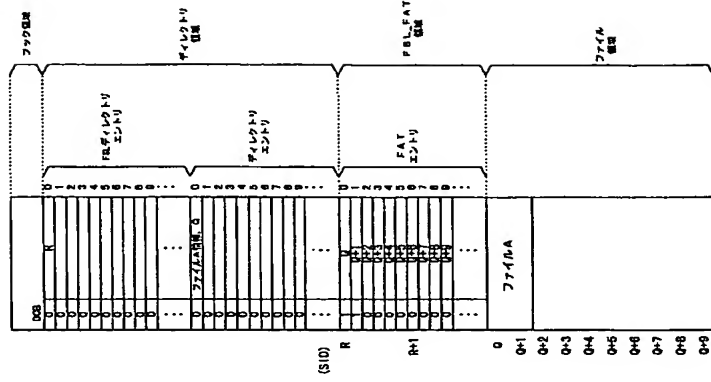
【圖15】



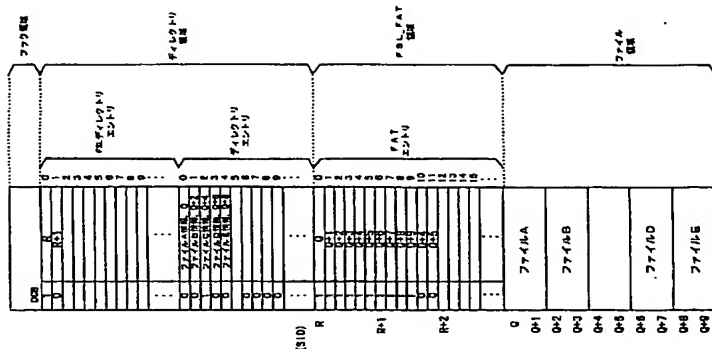
【図6】



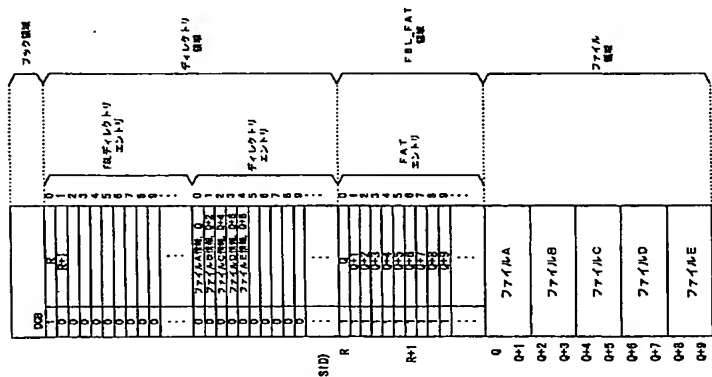
【図7】



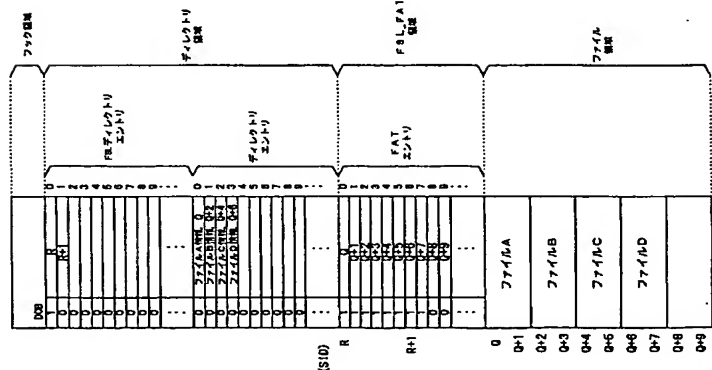
【図11】



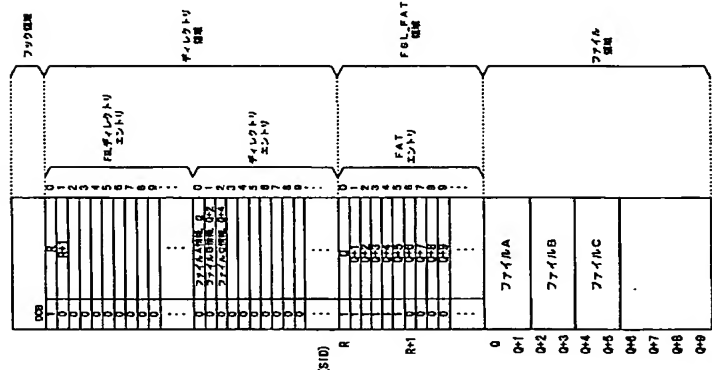
【図10】



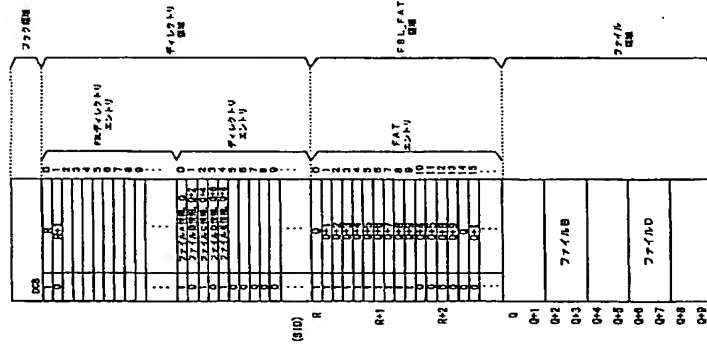
【図9】



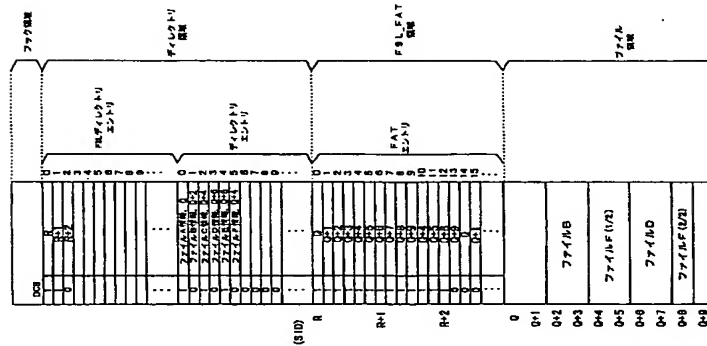
【図8】



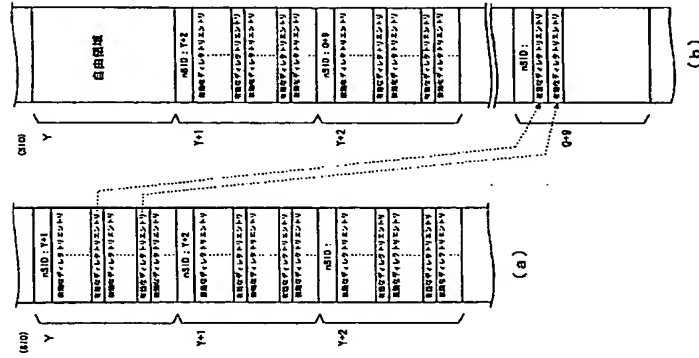
【図12】



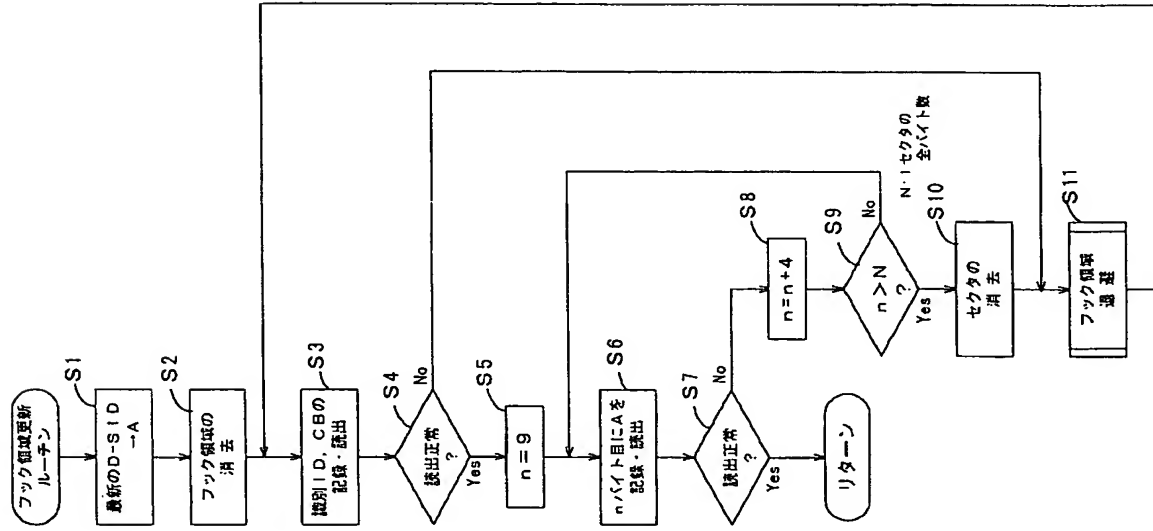
【図13】



【図16】



【図17】



【図18】

